1986-09

# Software requirements specification for an ammunition management system

Alderman, Robert Bruce

Monterey, California: U.S. Naval Postgraduate School

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


# THESIS

SOFTWARE REQUIREMENTS SPECIFICATION
FOR AN
AMMUNITION MANAGEMENT SYSTEM

by

Robert Bruce Alderman

September 1986

Thesis Advisor: B.A. Frew

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b RESTRICTIVE MARKINGS |
| 2a SECURITY CLASSIFICATION AUTHORITY | 3 DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution is unlimited. |
| 2b DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4 PERFORMING ORGANIZATION REPORT NUMBER(S) | 5 MONITORING ORGANIZATION REPORT NUMBER(S) |

| | | |
|---|---|---|
| 6a NAME OF PERFORMING ORGANIZATION<br>Naval Postgraduate School | 6b OFFICE SYMBOL (*If applicable*)<br>54 | 7a NAME OF MONITORING ORGANIZATION<br>Naval Postgraduate School |
| 6c ADDRESS (*City, State, and ZIP Code*)<br>Monterey, California 93943-5000 | | 7b ADDRESS (*City, State, and ZIP Code*)<br>Monterey, California 93943-5000 |
| 8a NAME OF FUNDING/SPONSORING ORGANIZATION | 8b OFFICE SYMBOL (*If applicable*) | 9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c ADDRESS (*City, State, and ZIP Code*) | | 10 SOURCE OF FUNDING NUMBERS |

| PROGRAM ELEMENT NO | PROJECT NO | TASK NO | WORK UNIT ACCESSION NO |
|---|---|---|---|
| | | | |

11 TITLE (*Include Security Classification*)
SOFTWARE REQUIREMENTS SPECIFICATION FOR AN AMMUNITION MANAGEMENT SYSTEM

12 PERSONAL AUTHOR(S)
Alderman, Robert B.

| 13a TYPE OF REPORT | 13b TIME COVERED | 14 DATE OF REPORT (*Year, Month, Day*) | 15 PAGE COUNT |
|---|---|---|---|
| Master's Thesis | FROM _____ TO _____ | 1986 September | 153 |

16 SUPPLEMENTARY NOTATION

| 17 COSATI CODES | | | 18 SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Ammunition Management, Ammunition Inventory Management, Automated Ammunition Management, Automated Ammunition Inventory Management. |
| | | | |
| | | | |

19 ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This thesis concerns the software requirements necessary to automate the present manual effort associated with ammunition inventory management and reporting at the afloat end-user level. Functional characteristics for the application software are developed, program and data structures are proposed and possible sources of data are identified.

The end-product of this research is the software requirements specification. This document supports further design development of the application software and is independent of programming language and system hardware configuration. The basic format satisfies the provisions of ANSI/IEEE Standard 830-1984.

| 20 DISTRIBUTION/AVAILABILITY OF ABSTRACT | 21 ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS | UNCLASSIFIED |

| 22a NAME OF RESPONSIBLE INDIVIDUAL | 22b TELEPHONE (*Include Area Code*) | 22c OFFICE SYMBOL |
|---|---|---|
| CDR B.A. Frew, SC, USN | (408) 646-2962 | 54FW |

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

# Software Requirements Specification for an Ammunition Management System

Robert Bruce Alderman
Lieutenant Commander, Supply Corps, United States Navy
B.S., University of Virginia, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1986

ABSTRACT

This thesis concerns the software requirements necessary to automate the present manual effort associated with ammunition inventory management and reporting at the afloat end-user level. Functional characteristics for the application software are developed, program and data structures are proposed and possible sources of data are identified.

The end-product of this research is the software requirements specification. This document supports further design development of the application software and is independent of programming language and system hardware configuration. The basic format satisfies the provisions of ANSI/IEEE Standard 830-1984.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

This research could not have been conducted without the assistance of many dedicated professional men and women. The author wishes to express special thanks to Ms. Rebecca Quarry of the Navy Ship's Parts Control Center, Mr. E. Eyanson of the Naval Supply Systems Command CAIMS Project Office, Mr. Ed Schlakta and Ms. Carol Cinque of the Naval Sea Systems Command FOSAMS Project Office, and Mr. James Dowling of the Defense Logistics Studies Information Exchange who prepared a custom bibliography for this effort.

Special appreciation is also extended to LCDR B.A. Frew and Professor T.P. Moore, both of the Naval Postgraduate School. These individuals suffered through innumerable drafts of this thesis to ensure its completeness and accuracy. Its success is due, in no small part, to their efforts.

Finally, my sincerest gratitude goes to my wife Patricia whose patience and understanding combined with an astute editing skill served to enhance the quality of an otherwise average work.

# I. INTRODUCTION

Fleet readiness is dependent on the effective management of materiel inventories. The logistics system which provides for provisioning and resupply of operating forces is extremely complex when viewed in the context of the Navy's global commitments and austere funding levels. Yet it is this logistics system which becomes all the more critical in times of national emergency. It is then when it becomes a life's blood, determining the range of deployment, endurance and even the tactics which can be employed.

Nowhere is this more evident than in the area of conventional ammunition management. The very nature of this commodity requires strict accounting and access to current information at all echelons of the Navy--a task made more complicated by the worldwide prepositioning of stock and the necessity for monitoring its serviceability. In response to this challenge the Conventional Ammunition Integrated Management System (CAIMS) was established.

The CAIMS is the Navy's central repository of ammunition inventory information. Program policy guidance for CAIMS is provided in [Ref. 1] with specific afloat policies implemented and further defined by Fleet Commanders. Administered by the Navy Ship's Parts Control Center (SPCC), CAIMS is designed to be the single point of reference in the Navy for information regarding the worldwide status of non-nuclear expendable ordnance [Ref. 2]. Accordingly, CAIMS performs multiple tasks and serves many users. Swanson [Ref. 3] notes that CAIMS is not only an inventory management tool, it is also used for readiness assessment; operational decision making; as a source of technical data;

for procurement, production and renovation planning; requirements determination; and in budget development.

Ammunition management is also big business. Navy ammunition procurement reached a high of $988 million during the Vietnam War [Ref. 3:p. 24]. In June 1980 the Navy's ammunition inventory was valued at $6.7 billion with $3 billion distributed to fleet units and overseas shore activities. These facts underscore the necessity for accurate and timely information concerning system inventory status: a major objective of the CAIMS. However, recent audits have questioned this system's ability to provide the required responsive support.

One such audit conducted by the U.S. General Accounting Office (GAO) [Ref. 4] has criticized the Navy's ability to maintain accountability over conventional ammunition. Based upon on-site audits of local records and comparison with data provided by the CAIMS, GAO found significant discrepancies between recorded data and actual on-hand assets. This was evident from a seventy percent error rate in account balances maintained by one naval magazine and by $8.5 million in gain and loss adjustments recorded between October 1979 and December 1980 by the same activity. The report concluded that the CAIMS data was unreliable and that the system is inadequate to maintain ammunition accountability. The report's recommendations are summarized below:

1. Develop a program to expedite the reconciliation of the Navy's central inventory records with storage records and investigate the causes of significant adjustments.
2. Develop a capability to effectively monitor the status of ammunition transactions.
3. Process suspended ammunition in a more timely manner. Suspended ammunition includes those items and components which are not ready for unrestricted use and that cannot be made serviceable using immediately available maintenance or repair capability.

4. Require interim accountability for ammunition designated for further transfer.

A subsequent GAO audit [Ref. 5] reconfirmed the need to implement these recommendations and reiterated the requirement for the Navy to improve its accountability and control over conventional ammunition. An in-house audit [Ref. 6] conducted by the Naval Audit Service of small arms and ammunition programs reached similar conclusions concerning the CAIMS inventory accuracy.

## A. PURPOSE

This thesis addresses the need to improve the interface between the CAIMS and the end-user. Specifically, it proposes a means to implement the GAO recommendations concerning timely and accurate transaction reporting and inventory reconciliation. The vehicle for achieving this is a system which automates the present manual recordkeeping and reporting functions at the afloat end-user level. This proposed system consists of data files and a software application program in a package termed the Ammunition Management System (AMS).

Toward this end, the thesis takes the form of a software requirements specification. Such a specification, according to Pressman [Ref. 7], establishes a complete information description, a detailed functional description, appropriate validation criteria, and other data pertinent to requirements. The software requirements specification defines the user's needs for the software component of an automated data processing system. It concludes the planning phase and further serves as the foundation for the subsequent development and maintenance phases of the software life cycle. This generalized software life cycle, as defined by Pressman, is depicted in Figure 1.1.

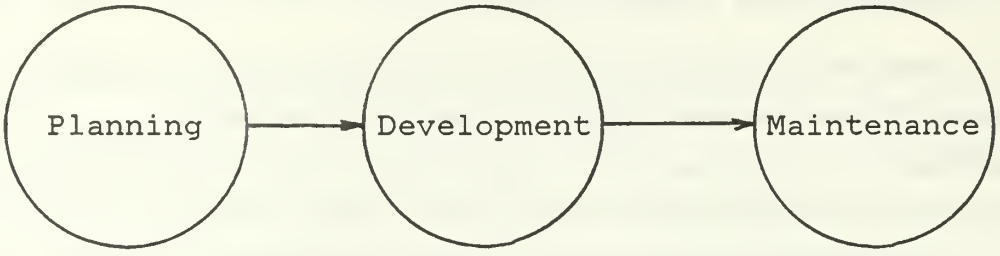



Figure 1.1
Generalized Software
Life Cycle

The common thread which binds the various phases together is user requirements. During the planning phase these requirements are identified and developed into characteristics of the desired software architecture. A translation then occurs during the development phase where a software product is formulated from the previously defined characteristics. The maintenance phase concludes the software life cycle and is evolutionary in nature. Here, the changes to user requirements drive the modification of the software product and ensure its currency and adaptability with the environment. This phase represents the most costly endeavor of the life cycle consuming up to seventy percent of an organization's software budget. Fifty percent of this has been directly attributable to perfective maintenance of the original delivered product [Ref. 7:p. 323]. This category of maintenance includes those actions to modify the software with new capabilities and general enhancements of initial capabilities in response to a changing environment and needs of the end-user. It is generally accepted that proper up-front research could reduce most of this cost. Therefore, the accurate determination of user requirements is intrinsic to the delivery of an effective and responsive software product.

Five questions have been formulated to assist this research effort:

1. What are the functional user requirements of the proposed software?
2. What are the required software design characteristics?
3. What are the data requirements to support the application software?
4. What are the validation criteria to test the proposed software?
5. What are the possible sources of data?

The answering of these questions, within the framework of software engineering, will not only serve to satisfy the requirements of the end-user but will also strengthen the system inventory management capability of the CAIMS. In this way data accuracy and reporting timeliness are enhanced at all levels of the CAIMS reporting structure.

## B. DISCUSSION

Naval units are required to maintain records and submit reports covering conventional ammunition inventories in their custody. These actions form the interface between the fleet end-user and the CAIMS. Records maintained onboard enable the management of onboard inventories of ammunition and support the requirement for external reports about onboard ammunition. Onboard records are standardized by [Ref. 2] and include such items as the Ammunition Lot/Location Card, Ammunition Master Stock Record Card and the Ammunition Serial/Location Card.

Reports, on the other hand, are tailored by Fleet Commanders to satisfy the unique reporting requirements of each fleet, in addition to satisfying the basic CAIMS data demands. These reports summarize data contained in records and are of a specialized nature. Such reports as the Ammunition Transaction Report (ATR), Maintenance Due Date

(MDD)/Missile Firing Expiration Date (MFED) Extension Requests and the Sonobouy Transaction Report (STR) are the primary status reporting means of the fleet. In addition, other supply documents such as requisitions, followups, modifiers and cancellations round out the necessary capabilities for ammunition inventory management.

Finally, as a closed-loop system, the CAIMS provides information to the end-user concerning the accuracy of user-generated reports and the material condition of onboard managed items. These include reconciliation reports originated by SPCC, and Notice of Ammunition Reclassification (NAR) messages originated by the systems commands. System effectiveness demands accurate input at the source. Accordingly, the onus for proper inventory status reporting begins with the end-user. Stated specifically:

> All CAIMS users have an obligation to pursue apparent errors in the CAIMS Data Base and ensure their reconciliation....To the extent that CAIMS data does not accurately reflect actual Navy assets, new ammunition procurements will not support fleet requirements. *It is vital to recognize that fleet support for ammunition is directly related to the timeliness and accuracy of fleet transaction reporting into CAIMS.* Therefore, accuracy in reporting cannot be emphasized too strongly . . . . [Ref. 2:p. 8-1-2].

This overriding concern for timely and accurate data input at the source is included as a designed-in objective of the proposed software. As discussed later, this concept is implemented by various features that provide the requisite accountability over ammunition inventory stocks.

## C. SCOPE OF THESIS

This research is limited to defining the software requirements of the end-user. Specifically, this thesis describes the application software necessary to automate and support ammunition inventory management and reporting at the

shipboard level. Accordingly, the unique requirements of ammunition load list management, as in the case of ammunition stores ships, is not addressed. A separate initiative in this area is the Fleet Optical Scanning Ammunition Marking System (FOSAMS) sponsored by the Naval Sea Systems Command. Interested readers are referred to [Ref. 8] for further project information. While the FOSAMS is not considered integral to this effort, interface requirements between the FOSAMS and the proposed software have been included in the functional description.

A second consideration concerns the environment in which the program will operate. For practicality it was decided to integrate the system into the existing inventory management and reporting structure and not attempt to design an independent system for this purpose. Accordingly, the Initial Operating Capability (IOC) will be limited in scope to the automation of current functions with the automated input and output resembling manual counterparts. In addition to achieving greater economy this action also reduces the need for retraining the ship's personnel. Flexibility is retained to allow program upgrades in subsequent releases. This will ensure program currency when procedures or policies change.

Finally, this effort covers only conventional ammunition management. The unique management requirements of weapons covered by various Navy Special Weapons Publications (Navy SWOPs) and included in the reporting structure of [Ref. 9] are not addressed. The exceptionally high security classification (Secret Restricted Data) assigned to these weapons, in addition to the low quantities of ammunition involved, does not lend itself to cost effective or secure automation.

## E. METHODOLOGY

This section establishes the framework for research and construction of the thesis. It is divided into three major areas covering the conduct of research, design approach and software metrics.

### 1. Conduct of Research

This research will follow generally accepted software engineering procedures with the end-product being a software requirements specification. The basic format of this thesis satisfies the provisions of [Ref. 10]. It is intended that this document serve as the basis for developing additional documentation to support follow-on design efforts on the proposed software. Accordingly, the onus is on identifying standard user requirements for an automated information system which is independent of software language or hardware configuration. In this way a more effective software design effort is supported. The proposed software may then be tailored, during the development phase, as either a stand-alone application or as an integrated subsystem in such standard ADP initiatives as the Shipboard Non-tactical ADP Program (SNAP). For this reason, *SNAP compatibility is a design goal of the proposed AMS.*

As previously mentioned, the software requirements specification defines the user's needs for the software component of an automated data processing system. In determining these needs an extensive search was conducted of the many sources concerning ammunition management.

This literature search was two-fold. First, a functional review of directives promulgated by Fleet Commanders and inventory managers was included to determine the existing inventory management and reporting policies.

Where possible, Navy training manuals and the author's personal experience served to supplement these procedures. The intent of this action was to take into consideration the "real life" or descriptive user environment. In this way a tempering of the software product is obtained thus realizing a higher probability of user acceptance.

Second, existing system deficiencies were noted by reviewing the previously mentioned audits. This approach takes a normative view of afloat ammunition management as it should be accomplished given the framework of established policies and procedures. In addition to automating the present manual procedures, it is highly desirable to correct as many documented discrepancies as feasible. Again, this emphasizes the importance of the end-user link to the CAIMS reporting structure. Objectives of this effort are to facilitate more efficient reconciliation of reported discrepancies, enhance transaction tracking by the inventory manager and provide for greater accountability of ammunition assets beginning with the end-user.

## 2. Design Approach

The software will incorporate certain engineering principles to ensure program validity and reliability. The purpose of this section is to address the minimum measures necessary to meet these objectives. The ancient adage of "An ounce of prevention is worth a pound of cure" has more applicability to software projects than many other endeavors. Mills notes:

> It is well known that you cannot test reliability into a software system. If programs are well designed in both data structure and control structure there is no contest between a programmer and a computer in finding errors: the programmer will win hands down . . . . So the first defense against errors is well designed programs and preventive proofing by authors themselves. [Ref. 11]

Therefore, it is appropriate to address these engineering principles and the method of incorporating them in the final product.

These principles have been derived by research efforts that are collectively referred to as "software engineering." Comer identifies the aim of software engineering as to improve the programmer productivity and increase the reliability, correctness, and cost effectiveness of the final product [Ref. 12:p. 169]. The application of software engineering principles requires an established methodology. This methodology, according to Pressman [Ref. 8:p. 15] is an approach using a set of techniques that are application-independent. He provides three key objectives of this effort:

1. A well-defined methodology that addresses a software life cycle of planning, development, and maintenance.
2. An established set of software components that documents each step in the life cycle and shows traceability from step to step.
3. A set of predictable milestones that can be reviewed at regular intervals throughout the software life cycle.

The fundamental building block of software engineering is the concept of program modularity. In addition to providing the means for implementing other design concepts, modularity enhances human understanding of the program logic. This latter view enables "intellectual management" [Ref. 7:p. 152] or "conceptual integrity" [Ref. 12:p. 268] of the software.

Modularity is one aspect of structured design. This approach, according to Stevens and others [Ref. 13] is a set of proposed general program design considerations and techniques for making coding, debugging, and modification easier, faster, and less expensive by reducing complexity. This is achieved by subdividing the software system. The

problem is decomposed into required functions and then refined ("stepwise refinement"). These functions are then translated into groupings of software code (modules) that are separately named and addressable. These elements are then integrated into a program structure to satisfy the problem requirements.

Modules may be characterized by "functional strength." This is where modules are designed to address a specific subfunction or task of the total requirements package. The measurement of the degree of functional strength of the module is called cohesiveness [Ref. 7:p. 158]. Complexity is reduced when modules have a high degree of cohesiveness. This allows for the concept of "information hiding" to be implemented whereby only data necessary for a given module to function is made available to that module. This data is "hidden" from other modules that do not have use of it. In this way program control paths, entry points and data availability are reduced with an increase in overall program independence.

Module cohesiveness also can impact memory efficiency and the speed of program execution. According to Peterson and Silberschatz [Ref. 14] a program is divided into "pages" which are loaded to memory "frames." These pages determine the locality of program execution.

> The locality model states that as a program executes, it moves from locality to locality. A locality is a set of pages which are actively used together . . . . A program is generally composed of several different localities which may overlap.
>
> For example, when a subroutine is called, it defines a new locality. In this locality, memory references are made to the instructions of the subroutine, its local variables, and a subset of global variables. When the subroutine is exited, the process leaves the locality, since the local variables and instructions of the subroutine are no longer in active use . . . . (L)ocalities are defined by the program structure and its data structures. The locality model states that all programs will exhibit this basic memory reference structure.

From this example it is evident that the more cohesive a module the higher the probability will be that necessary information will be in memory to support execution and the need to search for other pages is minimized.

Another qualitative criteria of module independence is coupling. Stevens and others provide the desired design objective in this area:

> The complexity of a system is affected not only by the number of connections but by the degree to which each connection couples (associates) two modules, making them interdependent rather than independent. Coupling is the measure of the strength of association established by a connection from one module to another. Strong coupling complicates a system since a module is harder to understand, change, or correct by itself if it is highly interrelated with other modules. Complexity can be reduced by designing systems with the weakest possible coupling between modules. [Ref. 13:p. 117]

More consideration of coupling will be provided in the internal interface section. For now the previous discussion is adequate to continue the examination of other software engineering principles.

With the proper construction of individual modules ensured by adherence to cohesion and coupling objectives, we can now attend to design of an integrated program. This section concerns the design topology of the program structure. Methods of integration are discussed later in the validation criteria chapter.

Program structure denotes hierarchical control from the top-down. Control relationships may by depicted in a box diagram, such as Figure 1.2, where each box represents an independent module. In this diagram, control is "factored" down from superior to subordinate modules. Pressman [Ref. 7:p. 149] mentions that in this way design and implementation are simplified, testability is enhanced, and maintenance can be approached in a more efficient manner.

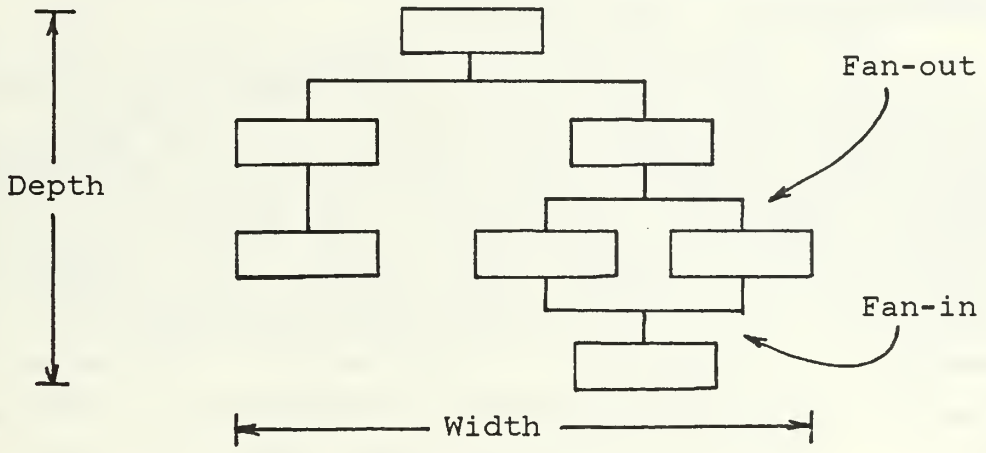



Figure 1.2
Program Structure Diagram
(Example)

While Pressman stresses that there is no single correct approach to factor control in a program, he does provide eight design heuristics, or guidelines, that enable successful design. He further notes that application of these heuristics is independent of a specific design methodology. [Ref. 7:pp. 169-174]

1. Evaluate the preliminary software structure to reduce coupling and improve cohesion.
2. Attempt to minimize structures with high fan-out; strive for high fan-in as depth increases.
3. Keep scope of effect of a module within the scope of control of that module. As an example, if a variable's value is changed during module execution, the results of that effect should be limited to those modules under the control of the module making the modification.
4. Evaluate module interfaces to reduce complexity and redundancy and improve consistency.
5. Define modules whose function is predictable, but avoid modules that are overly restrictive.
6. Strive for single-entry, single-exit modules, avoiding "pathological connections" (i.e., multiple entry points).
7. Package software based on design constraints and portability requirements.
8. Select the size of each module so that independence is maintained.

These heuristics have been incorporated into the planning effort of this project.

The treatment of design approach in this section has been intentionally cursory in nature. The intent was to address major software design concerns which can affect the planning phase and not bog down in the details of the various divergent views on this subject. In the next chapter the concept of data flow-oriented design is presented. The objective of this method is to derive a software architecture through the translation of information flow.

## 3. Software Metrics

The previous section discussed preventive measures that are to be designed into any viable software program. However, these methods, in themselves, do not provide an indication of the resulting program complexity which can affect such things as development cost and process efficiency. Both of these have major impact on the ultimate success of the software. To provide a more complete picture of the software, various software metrics have been developed. In this section we will discuss representative metrics and their application to the program at hand.

A metric is defined as a measurable indication of some quantitative aspect of a system. DeMarco lists five such quantitative aspects requiring measurement in a typical software project. These are scope, size, cost, risk and elapsed time. He further allocates metrics into one of two categories as either a result or as a predictor. [Ref. 15:pp. 49,50] In the development phase of the software life cycle we are more concerned with the use of metrics to predict and enhance the productivity of the software development effort. The wealth of literature on software

development relates to the estimating of software productivity effort. Not surprisingly, the management tools available for this purpose are extensive [Ref. 16]. However, the availability of metrics to predict software quality are more elusive [Ref. 7:p. 164]. Of these metrics, the cyclomatic complexity measure and software science show the most promise, albeit still in their infancy. Both of these methods satisfy the major functions of a software metric as defined by Curtis [Ref. 17]:

1. Serve as a management information tool.
2. Serve as a measurement of software quality.
3. Provide feedback to the software engineer.

The first metric is the cyclomatic complexity measure proposed by McCabe [Ref. 18]. His efforts serve to answer the question: "How to modularize a software system so the resulting modules are both testable and maintainable." The metric he develops uses the number of control paths through a program as a measure of complexity. For example, a program segment is represented as a process graph (G) in planar space and is depicted in Figure 1.3. The cyclomatic number v(G) is the effective metric computed by the formula:

$$v(G) = e - n + p$$

where e is the number of edges, n is the number of vertices of the graph, and p is the number of connected components. The nodes of the graph represent modules of software code. For the graph in Figure 1.3 v(G) is equal to 4. This is computed from the above formula as follows:

$$v(G) = 7 - 5 + 2 = 4$$

For a strongly connected graph (with unique entry and exit nodes) v(G) is equal to the maximum number of linearly independent circuits. Stated another way, the cyclomatic metric may be computed by counting the enclosed regions and adding one for the surrounding area.

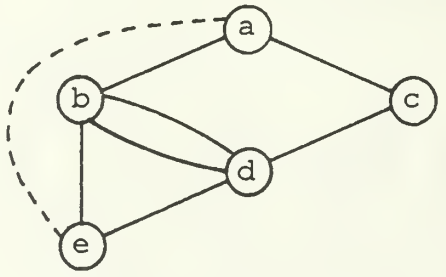



Figure 1.3
Process Graph
(Example)

McCabe describes the application of this metric as follows:

> The overall strategy will be to measure the complexity of a program by computing the number of linearly independent paths v(G), control the "size" of programs by setting an upper limit to v(G) (instead of using just physical size), and use the cyclomatic complexity as the basis for a testing methodology. [Ref. 18:p. 309]

Based on experience gained from observing programmers involved in differing software projects, McCabe set this upper limit at ten. This figure was based more in reasonableness rather than magic. Although the intent was to limit the size of modules and allow for testing of all independent paths, this approach had an additional positive affect. The metric enforced a discipline on the programmer to follow structured programming rules. McCabe noted that

even programmers who had no formal training in structured programming consistently produced code in the 3 to 7 complexity range.

At this point one may wonder why we are even discussing McCabe's complexity measure since it deals with modules of code and the actual coding doesn't begin until the development phase of the life cycle. The answer lies in the need to predict and limit the program complexity and properly identify resource requirements early-on. The concept that enables application of the complexity metric at this point is the abstract process.

Mekly and Yau [Ref. 19] define an abstract process as a representation schema for a discrete phase of system activity directly associated with some function and identifiable by the initial and final states with respect to that function. Pressman expands this to include various views of the same system:

> When we consider a modular solution to any problem, many levels of abstraction can be posed. At the highest level of abstraction, a solution is stated in broad terms, using the language of the problem environment. At lower levels of abstraction, a more procedural orientation is taken . . . .
>
> Each step in the software engineering process is a refinement in the level of abstraction of the software solution . . . . (T)he lowest level of abstraction is reached when source code is generated. [Ref. 7:pp. 154,155]

The concept of the abstract process supports development of abstract process networks (AP-nets) which, in their basic sense, serve as "software blueprints." The AP-net is a representation of the software system and indicates operations on system control and data transformation. Mekly and Yau [Ref 19:p. 431] note that the "orthogonality" of control and data flows in an abstract process allows AP-net use to represent system characteristics in terms of either control or data flow. This observation has permitted the

application of the complexity metric in designing Data Flow Diagrams (DFDs). The DFD is essentially a graphic tool used to depict information flow characteristics. The application of this technique is discussed in the next chapter.

The second area of software metrics has been proposed by Halstead [Ref. 20] and is called software science. This method provides a highly quantitative measurement approach which views software from many perspectives including program length, volume, level and purity. Although the major thrust of this work is result (vice predictive) oriented, it is included for its potential use as a formal measure of program size and resulting complexity. As such, it can be used to develop a design approach in the planning phase. In the development phase software science can assist in the selection of a target language which maximizes the efficiency of implementation for a given application.

The effective metric for this purpose is called the program level (L). It has its genus in the following:

> Intuitively, the concept of the level at which a program might be written has been with us since the first "Higher-Level Languages" were referred to as such. Before a concept of this type can have much scientific utility, however, it must be reduced to quantitative or measurable terms . . . . (O)nce a given algorithm and a given language are decided on, alternative implementations may be comparatively ranked only on the basis of expert opinion, or perhaps by the opinionated experts. Yet it is quite true that the level of implementation is vitally important in programming, because it contributes to the effort of writing, propensity for error, and ease of understanding. [Ref. 20:p. 25]

The program level of the implementation of an algorithm is defined as:

$$L = 2/\eta_1 \times \eta_2/N_2$$

where the primitive measure $\eta_1$ is the number of unique operators that appear in an algorithm; $\eta_2$ is the number of unique operands that appear in an algorithm; and $N_2$ is the total number of operand occurances. From this relationship a tradeoff may be determined and a language selected that is optimal to the application at hand. A language which decreases the number of unique operands in relation to the number of unique operators (for a given application) would result in a lower program level. The "algorithm" for our purposes would be the data flow diagram, again applying the concept of abstraction.

In theory, this algorithm must be capable of implementation in some minimum volume. In this case, the program level equals one and represents the most efficient implementation feasible. A caveat must be introduced at this point, however. Effective usage of the program level metric in the planning phase requires that reliable data be available from sample implementations of similar algorithms. Only in this case can alternative algorithms be compared and an implementation strategy selected.

The program level metric is not operational in the planning phase, per se. The basic reason for this is that implementation is not an objective of this phase. This metric does serve an important planning function, however, and deserves mention here. The program level forces consideration of the design requirements of the following development phase. In so doing, simplicity and efficiency are introduced at an early stage of requirements planning. This is reflected in the economy of the data flow diagrams presented later (i.e., minimizing the number of nodes per level) and will pay off in easier coding, testing and maintenance later on.

## II. INFORMATION DESCRIPTION

During the planning phase, user requirements are identified and then translated into desired software characteristics. This process results in the definition of functional program capabilities and the necessary software architecture. This chapter concerns the first of two intermediate steps essential in this transformation process. This is the analysis of information flow. The software engineering methodology for this purpose is a process called data flow-oriented design. The objective of this method, according to Pressman, is to provide a systematic approach for the development of software structure, an architectural view of software and the underpinning of the preliminary design step [Ref. 7:p. 178]. He further notes that this transition from information flow to structure is realized in a five step process as follows [Ref. 7:p. 180]:

1. Information flow category is established.
2. Flow boundaries are indicated.
3. Data flow diagrams are mapped into software architecture.
4. Control hierarchy is defined by factoring. The term "factoring" means distributing control among software modules from the top-down.
5. The resulting structure is refined by the use of design measures and heuristics.

These steps are performed by first deriving the data flow diagrams and data structures necessary to support graphic depiction of the information flow. Secondly, a data dictionary is provided to describe the data environment of the software and to establish standards for data element representations or definitions. A data acquisition strategy is then proposed. In keeping with the previously stated

design goal of compatibility with the SNAP, the proposed method of data acquisition follows the established data draw-down and build procedures. Finally, a program structure is derived with a control hierarchy factored among independent modules.

## A. DATA FLOW DIAGRAMS

As previously mentioned, the DFD is a graphic tool used to depict information flow. The DFD occupies an invaluable place in most software engineering methodologies. It is this building block that is used to map the desired software structure into the data flow-oriented design discussed later. In addition, by applying the concept of AP-nets to the DFD, an incremental refinement may be accomplished for process representations beginning with the highest level representation and continuing down through the lower levels.

The construction of the DFD is relatively simple and requires only four constructs. These are summarized as follows:

> Information (i.e.: data flow) is represented by a labeled arrow. Processes (transformations) are represented by appropriately labeled bubbles. Information sources and sinks are noted as labeled boxes, and stored information (e.g.: a data file) is represented by a double horizontal line. An information source is a location where data originates . . . . An information sink is the final destination of data as it moves through the system. [Ref. 7:p. 101]

Pressman [Ref. 7:p. 101] notes three attributes of a DFD:

1. Information flow in any system- manual, automated, or hybrid- can be represented.
2. Each bubble (node) may require significant refinement to establish complete understanding.
3. Flow of data, rather than flow of control, is emphasized.

The last attribute, flow of data, is an important concept at this point. The DFD only displays logical processes and does not indicate control hierarchy. The DFD is related to program structure, however, through the previously mentioned mapping process. This process is the translation of the flow of data (represented by the DFD) into a control hierarchy (represented by software structure diagrams).

For the application at hand, the analysis begins with the Fundamental System Model depicted in Figure 2.1. This is the most basic level of abstraction where the entire system function is represented by a single node, or information transform. The "black box" approach provides for a system overview of information inputs and product outputs. In addition, it serves as an intellectual starting point for subsequent refinements to the system.

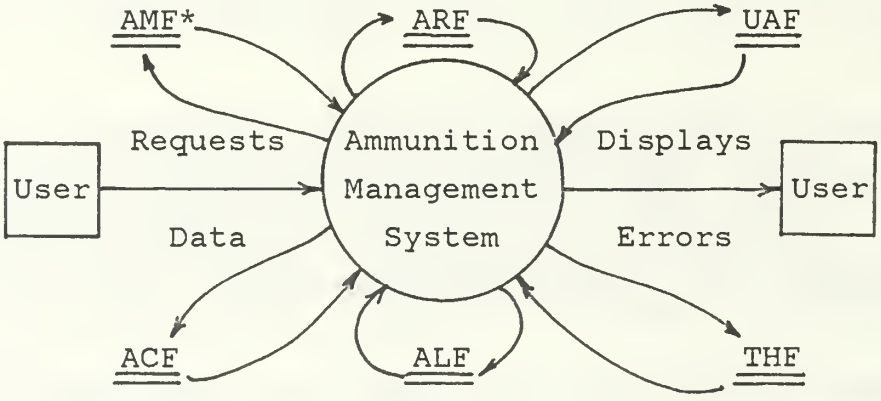



Figure 2.1
Fundamental System
Model

*File descriptions are provided by logical record formats in Table 1 on page 41.

Appendix A provides data flow diagrams which are refinements of higher level models. That is, the Level 1 model is a refinement to the Fundamental System Model, Level 2 models are refinements to Level 1 models, and so forth. The logical boundary of a process that is included in a given refinement is called the "domain of change."

The domain of change supports and builds on the concept of the abstract process. In this way, high level models need give only cursory detail of process functions and data flows. These models may later be refined, within a domain of change, as user requirements are clarified or when pending modifications are implemented. The rules for model refinement are governed by the characteristic of the process being represented by the node. A process may be characterized as either a data transaction (i.e., Select Function) wherein data is changed as a result of a triggering action, or as a data transform wherein data is modified along a path over a period of time.

The distinctions between transaction and transform analysis will not be included here. The reader is referred to [Ref. 7:pp. 182-197] for an intensive handling of this subject. Some general guidelines to be followed during model construction and refinement are [Ref. 7:pp. 102-104]:

1. The first data flow diagram layer should always be the fundamental system model.
2. Primary input/output (I/O) files should be carefully noted.
3. All arrows and (nodes) should be labeled (with meaningful names).
4. Information continuity must be maintained. That is, input and output to the refined model must remain the same as in the original model.
5. One (node) at a time should be refined.

## B. DATA STRUCTURE REPRESENTATION

A data structure may be informally defined as an organized collection of values and a set of operations on them [Ref. 21]. A data model is an extension of this concept. It provides a method to organize, represent, access, store, modify and process the data. Sprague and Carlson [Ref. 22] identify five data models of which four may be used in this effort.

We will discuss the record and relational models in this section. The former is the oldest and most common approach to data organization, the latter being the most state-of-the-art.

The record model is the traditional approach to data organization and has found wide acceptance and use in business and military non-tactical applications. In this model, data is organized into files in order to support specialized application programs. This scheme is depicted in Figure 2.2. For the proposed AMS we would require four separate files with associated application programs. The files contain records, which in turn are subdivided into fields. The records are identified by one or more fields, called keys, which contain unique values, i.e., different values for each different record.

Although this is a straight forward approach to processing, it is susceptible to what is referred to as data modification anomalies. As an example, if it was necessary to add a field to the Ammunition Management File, the associated inventory program would have to be changed and updated. But the problem does not end there. The Ammunition Requisition File and program would also have to be updated to reflect this change even though the data field may not be used in that processing activity.

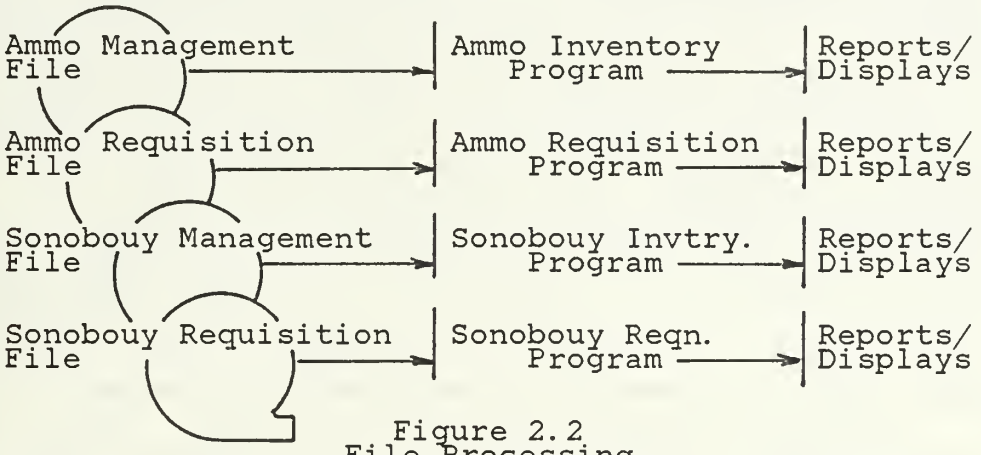



Figure 2.2
File Processing System

Another problem occurs when data is lost. This is called a deletion anomaly. If, for example, the unit price of an item is only shown in the Requisition File, and the only outstanding requisition for an item is received or cancelled, we lose the unit price data for that item.

In response to the data modification anomaly problems of the record model, the relational model was developed. In this scheme, data is organized into files according to rules called "normal forms." There are currently seven such normal forms identified [Ref. 23], however, most data design efforts are limited to satisfying the first three. These are:

1. A relation cannot have any repeating groups (fields).
2. Attributes (fields) must relate to the primary key.
3. Attributes must only relate to the primary key and not to any other field.

The higher level normal forms are not used because it would require more money to implement them than is required to accommodate the anomalies which would remain.

In the relational model, data is organized into "relations." The relation is a construct of "attributes," or fields, that are functionally dependent on the primary key. It is this concept which provides the much needed independence between files and the associated reduction of modification anomalies. The discussion of relational database theory goes beyond the scope of this paper. A good overview of this topic is provided in [Ref. 24] and [Ref. 25]. In addition, the reader is referred to [Ref. 26] for a discussion of the normal forms and their application.

A side effect of using the first three normal forms is that they tend to proliferate relations. In fact, the number of relations increase significantly with each attempt to incorporate a higher level normal form in the data base design. The motivation for this, however, is that the relational data base management system offers greater data accessibility and flexibility. Through the mathematics-based set operations characteristic of the relational data base system more data is made available to the user and greater efficiency is provided over standard file processing systems due to the reduction in data redundancy. Figure 2.3 demonstrates this capability.

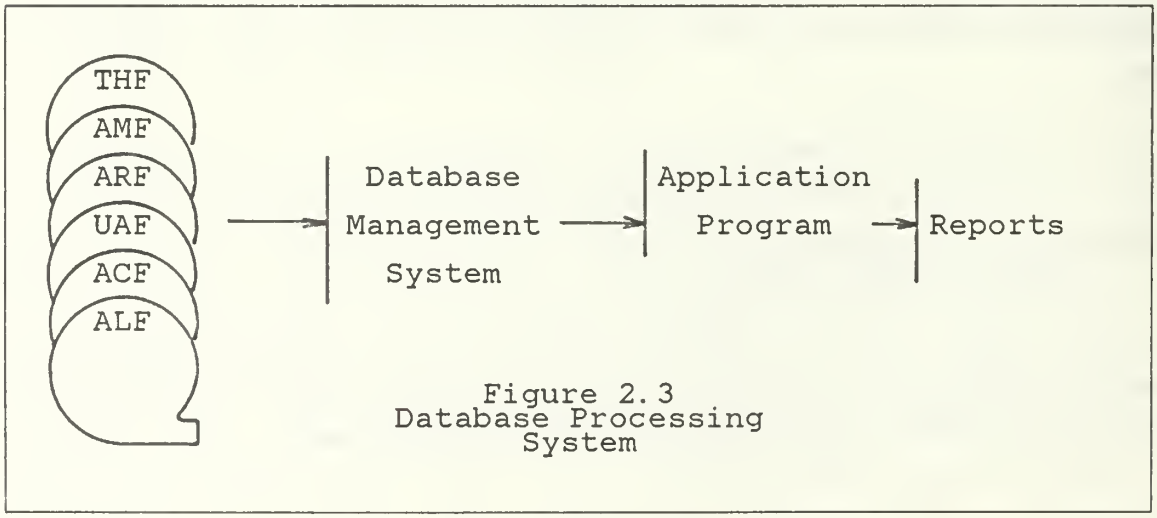



Figure 2.3
Database Processing System

The files shown in Figure 2.3 are described by their logical record structures in Table 1 (p. 41). In order to retain flexibility in describing the data structures, these records are presented in a "pseudo-COBOL" hierarchical structure. This enables better understanding of the record contents and data relationships. In addition, this approach permits easy translation into COBOL record format as well as database relations.

Provisions for COBOL record format translations must be made. First, COBOL is presently the Navy's standard business computer language. Second, this approach facilitates enhanced data understanding by systems analyst and programmer personnel, a majority of whom are COBOL literate.

The logical records presented in Table 1 satisfy the first three normal forms with the exception of the Ammunition Constants File (ACF). The unique application of this file does not lend itself to normalization beyond the first normal form. Update anomalies are avoided, however, in that only one record resides in the file and that data elements are not found in other files.

C. DATA DICTIONARY

The data flow diagrams provide a blueprint of information flow in terms of data transformation and transaction. The data description proposes a logical data structure to support this process. The purpose of this section is to define the data elements themselves. The vehicle for this is the data dictionary.

The data dictionary provides meaning to the data represented in the data flow diagrams. It defines data elements and provides such characteristics as allowed values (codes, etc.), aliases, supporting references, and

identifies user programs. In addition, as a design document, the data dictionary serves as an important project reference that allows standardization at an early phase of the life cycle. This capability enables program portability and easier maintenance.

It is this last reason that the Standard Data Element Dictionary (SDED) [Ref. 27] was developed. The SDED is published by the Navy Fleet Material Support Office as a reference document for designers of uniform automated data processing systems, systems analysts, and programmers for identifying and obtaining COBOL descriptions of data elements used in NAVSUP managed data processing systems. The CAIMS is included within its scope.

In keeping with the emphasis for standardization, the AMS data dictionary (Appendix B) utilizes standard data element names, where applicable, from the SDED. Local data elements are defined using the SDED entry format in Table 2 (p. 46). The data dictionary is a dynamic document and will require frequent revision during the development phase as new elements are identified. In addition, The AMS data dictionary only contains those data elements included in the logical record structures. Global and local program variables that are not associated with a logical record also require definition. These should be entered prior to actual coding, preferably during the process narrative ("pseudocoding") step of the development phase.

## D. DATA ACQUISITION

The acquiring, formatting and integration of data can be an expensive and time consuming task. The extent of this effort cannot usually be determined a priori. This is due to the fact that there is no standard software implementation to provide a benchmark. There is, however,

general agreement that the existence of data prior to development significantly reduces both cost and effort involved.

Sprague and Carlson [Ref. 22:pp. 223-225] provide an example of this effect in the implementation of a decision support system. He noted that in those implementations where preexisting data was not available the cost of data acquisition amounted to over fifty percent of implementation costs with ten percent of manhours devoted to data validation or correction. These same figures were reduced to less than ten percent of implementation costs and less than one percent of manhours when a preexisting data base was available. These facts present a strong argument for the use of established data repositories.

For the AMS project, this repository is the current CAIMS data base and the local records maintained by the ship. Such an approach is not new. CAIMS summary reports are presently available to inventory manager and Fleet staffs by Navy Ammunition Logistics Code (NALC) or DOD Item Code (DODIC) [Ref. 28]. This information includes balance quantities (serviceable/unserviceable), allowance, and monthly and cumulative expenditures by type (combat, training, etc.). A "draw-down" of this data could be conducted and a data base constructed that is tailored to a specific ship. This approach parallels the data acquisition strategy for the SNAP II where the Weapon Systems File is used to construct shipboard data configuration files. The ability to integrate these two draw-downs is possible. Such an accommodation would allow the AMS to be implemented as a subsystem in the SNAP II. Figure 2.4 (pp. 39,40) depicts the CAIMS/local data acquisition overlayed with other SNAP II data. The local data provided by the ship would augment that provided by the CAIMS draw-down and include that data

which is not available such as responsible work center and location.

In addition to simplifying the data acquisition process, this strategy would have the added benefit of protecting the integrity of the CAIMS as the sole repository for all ammunition stock status. From lessons learned in numerous SNAP II installations, errors are best identified and corrected at the end-user level. Following implementation, the ship should be required to conduct a review of its CAIMS reported allowances and stock balances. Automatic ATR/STR documents could then be produced incident to data correction in a way similar to the way that configuration change reports are now produced by SNAP II. This is implemented by protecting the various records through the application software. If changes are made by users to these records, and if such changes fall under externally reportable criteria, the software flags these changes and includes them in subsequent report generations. This process is conducted automatically and is invisible to the user.

## E. PROGRAM STRUCTURE

The derivation of the program structure is a major objective of the planning phase. This translation from the data flow diagrams, presented earlier, operationalizes the design heuristics as they pertain to functional cohesiveness and coupling, and to factoring of control and modularity. These structures are provided in Appendix C.

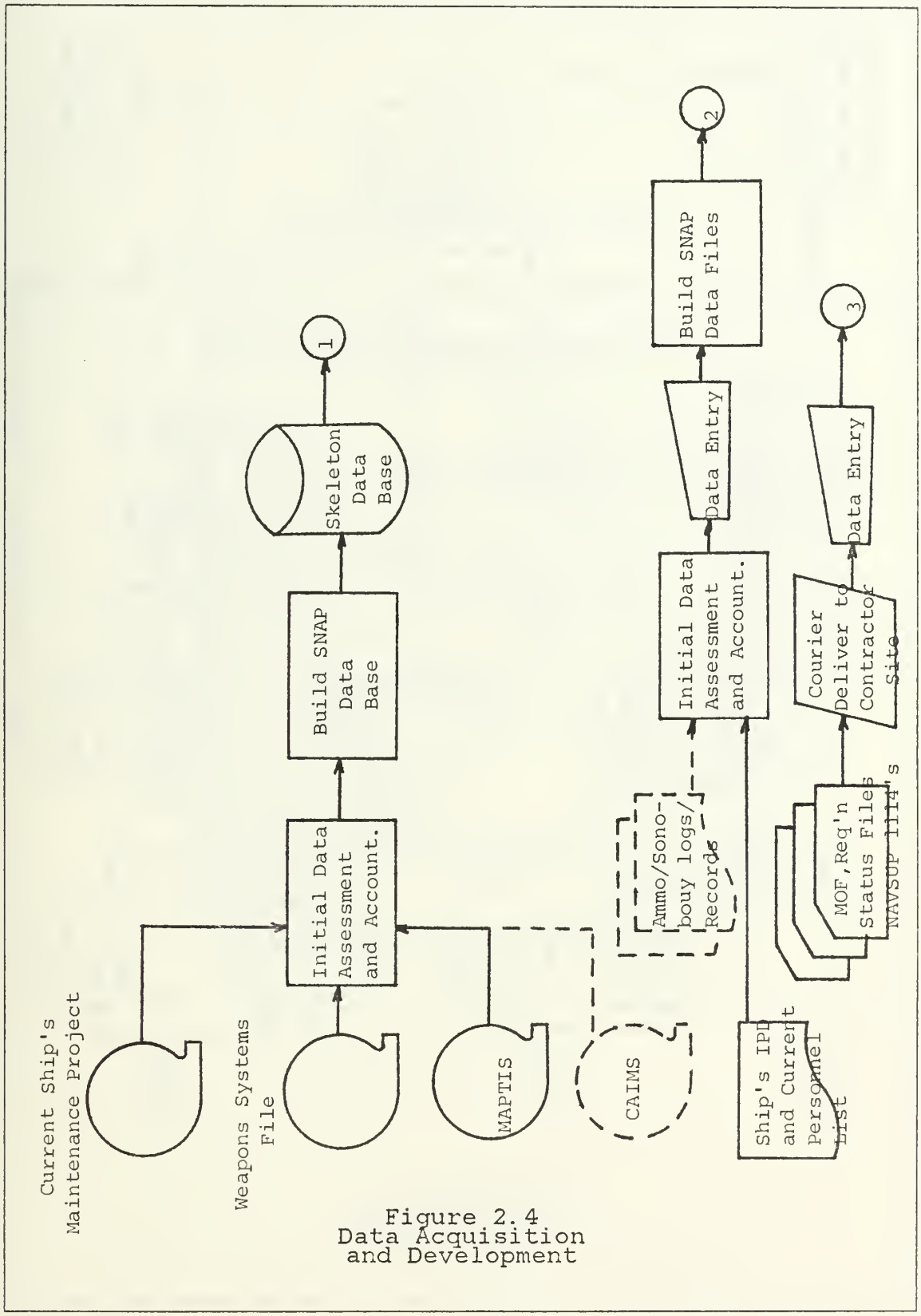



Figure 2.4
Data Acquisition
and Development

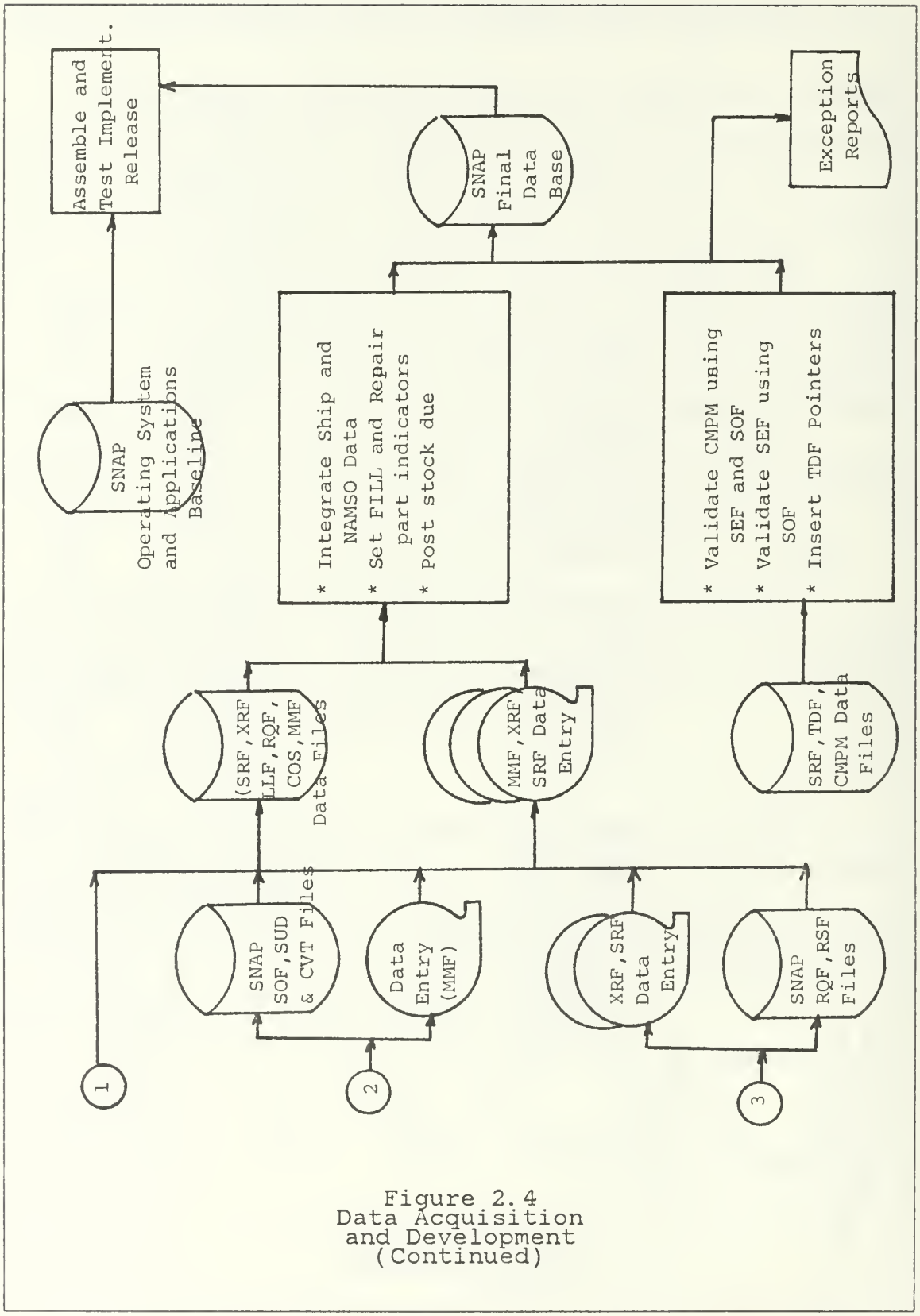



Figure 2.4
Data Acquisition
and Development
(Continued)

TABLE 1

LOGICAL RECORD STRUCTURES

Ammunition Management File (AMF)
Record Structure*

```
01 Ammo-record.
** 02 DODAC.
      03 Federal-supply-classification   char (4).
      03 DOD-ammo-code-or-NALC           char (4).
   02 Edit-lock                         char (1).
   02 Ammo-alowc-list-nr                num (5).
   02 Resp-work-center                  char (4).
   02 National-item-idfcn-nr            char (9).
   02 Nomenclature                      char (30).
   02 Nick-name                         char (10).
   02 Cog-symbol-requisitioned          char (2).
   02 Unit-of-issue                     char (2).
   02 Fund-code                         char (2).
   02 Allow-qty                         num (5).
   02 Total-DODAC-qty-onhand            num (5).
   02 Reorder-qty                       num (5).
   02 Due-quantity                      num (5).
   02 Reorder-flag                      char (1).
   02 Unit-price                        num (10).
   02 Maint-due-date                    num (4).
   02 Shelf-life-info.
      03 Shelf-life-code                char (2).
      03 Shelf-life-action-code         char (2).
   02 Last-update.
      03 User-id                        char (3).
      03 Change-date                    num (4).
```

* One record per allowed DODAC.
** Primary key.

TABLE 1

(Continued)

Ammunition Requisition File (ARF)
Record Structure*

```
01 Reqn-record.
** 02 Document-number.
      03 Document-julian-date                num (4).
      03 Document-nr-serial-nr               char (4).
   02 Edit-lock                              char (1).
   02 DODAC.
      03 Federal-supply-classification       char (4).
      03 DOD-ammo-code-or NALC               char (4).
   02 Reqn-outstanding-flag                  char (1).
   02 Cancellation-flag                      char (1).
   02 Cancellation-sent-date                 num (4).
   02 Followup-flag                          char (1).
   02 Followup-sent-date                     num (4).
   02 Document-identifier                    char (3).
   02 Activity-routing-identifier            char (3).
   02 Media-and-status-code                  char (1).
   02 Order-quantity                         num (5).
   02 Demand-code                            char (1).
   02 Signal-code                            char (1).
   02 Distribution-code                      char (1).
   02 Project-code                           char (3).
   02 Priority-code-other-cog                num (2).
   02 Required-delivery-date                 char (3).
   02 Advice-code                            char (2).
   02 Exception-flag                         char (1).
   02 Exception-info                         char (30).
```

(Continued on next page)

TABLE 1
(Continued)

```
02 Status-info.
   03 Status-code                          char (2).
   03 Rtng-idr-last-holding-actvy          char (3).
   03 Estimated-shipping-date              num (3).
   03 Status-date                          num (3).
02 Partial-order-info.
   03 Receipt-date                         num (4).
   03 Received-quantity                    num (7).
   03 Balance-quantity                     num (8).
   03 Balance-ESD                          num (3).
02 Last-update.
   03 User-id                              char (3).
   03 Change-date                          num (4).

 * One record per outstanding requisition
** Primary key.
```

User Access File (UAF)
Record Structure*

```
01 User-record.
   02 Access-vector.
   ** 03 User-id                  char (3).
      02 Password                 char (6).
      02 Work-center-code         char (4).
      02 Access-code              num (1).
   02 User-name                   char (15).
   02 Rank                        char (5).
   02 Phone                       char (4).
   02 Last-update.
      03 User-id                  char (3).
      03 Change-date              num (4).

       * One record per user
      ** Primary key.
```

TABLE 1

(Continued)

Ammunition Constants File (ACF)
Record Structure*

```
01 Constants-record.
   02 Unit-info.
      03 Unit-name                      char (20).
      03 Hull-number                    char (6).
      03 Rqnr-identification-code       char (6).
      03 Unit-PLAD                      char (15).
   02 TYCOM-PLAD                       char (25).
   02 TYCOM-act-info-code              num (1).
   02 IUC-PLAD                         char (20).
   02 IUC-act-info-code                num (1).
   02 ISIC-PLAD                        char (20).
   02 ISIC-act-info-code               num (1).
   02 Addee-01-PLAD                    char (20).
   02 Addee-01-act-info-code           num (1).
   02 Addee-02-PLAD                    char (20).
   02 Addee-02-act-info-code           num (1).
   02 Addee-03-PLAD                    char (20).
   02 Addee-03-act-info-code           num (1).
   02 Addee-04-PLAD                    char (20).
   02 Addee-04-act-info-code           num (1).
   02 Addee-05-PLAD                    char (20).
   02 Addee-05-act-info-code           num (1).
   02 Addee-06-PLAD                    char (20).
   02 Addee-06-act-info-code           num (1).
   02 Expend-approving-auth            char (15).
   02 Ship-to-info.
      03 Ship-to-UIC                   char (6).
      03 Ship-to-name                  char (15).
   02 First-destination-info.
      03 First-dest-UIC                char (6).
      03 First-dest-name               char (15).
```

* One record per ACF.

TABLE 1

(Continued)

Ammunition Lot File (ALF)
Record Structure*

```
01 Lot-record.
** 02 Ammunition-lot-number                  char (16).
   02 Edit-lock                              char (1).
   02 DODAC.
      03 Federal-supply-classification       char (4).
      03 DOD-ammo-code-or-NALC               char (4).
   02 Location                               char (10).
   02 Receipt-date                           num (4).
   02 Received-quantity                      num (7).
   02 Total-lot-qty-onhand                   num (5).
   02 Condition-code                         char (1).
   02 Maint-due-date                         num (4).
   02 Expenditure-pending-flag               char (1).
   02 Last-update.
      03 User-id                             char (3).
      03 Change-date                         num (4).
```

* One record per lot or serial number.
** Primary key.

Transaction History File (THF)
Record Structure*

```
01 Transaction-record.
** 02 Document-number.
      03 Document-julian-date                num (4).
      03 Document-nr-serial-nr               char (4).
   02 DODAC.
      03 Federal-supply-classification       char (4).
      03 DOD-ammo-code-or-NALC               char (4).
   02 Transaction-type-code                  char (1).
   02 Activity-routing-identifier            char (3).
   02 Transaction-info.
      03 ATR-trnsn-qty                       num (9).
      03 ATR-trnsn-code                      char (1).
      03 ATR-trnsn-flag                      char (1).
      03 ATR-trnsn-date                      num (4).
      03 ATR-condition-code                  char (1).
   02 NAR-nr                                num (5).
   02 NAR-srl                               num (3).
```

* One record per requisition or expenditure document number.
** Primary key.

TABLE 2

DATA DICTIONARY ENTRY FORMAT

| | |
|---|---|
| DEN: | (Data Element Number) For those data elements that are cataloged in the Standard Data Element Dictionary (SDED) a DEN is assigned. The DEN consists of an alphabetic character followed by three or four numeric characters. The DEN acts as a means for controlling data elements and as a shorthand name. |
| TITLE: | The title is the actual name given to the data element. It may contain up to 60 characters. |
| COBOL: | The unique standard COBOL data name assigned to the data element is provided here. COBOL names conform to the rules of NAVSUP Publication 507 and may contain up to 30 characters including hyphens. |
| PIC: | The COBOL picture specified for an element is the standard for exchanging that data between systems. |
| DESC: | The narrative description precisely explains what data or information the data element represents. |
| NOTES: | Information not directly related to the meaning of the data element but of interest to users is entered under NOTES. |
| REFS: | If another publication or document contains additional information on an element it is included under REFS. |
| CODES: | If the data is of the form of codes all codes and their definition are listed under CODES. |

TABLE 2

(Continued)

| | |
|---|---|
| ORIG: | The organizational code of the analyst originating the data element definition is provided under ORIGINATOR. |
| CREATED: | If the data element is cataloged in the SDED, the day, month and year in which the definition was updated is included under CREATED. |
| UPDATED: | This section includes the day, month and year in which the SDED data element was updated. |
| USER: | This section identifies the various system users of a particular data element. |

# III. FUNCTIONAL DESCRIPTION

The information description presented in Chapter 2 provided one view of the proposed software from the standpoint of data. In this approach, the system was represented by an information flow consisting of data transformations and transactions. This blueprint was then translated into a program architecture to support software coding and verification in the subsequent development phase.

Another view must also be included to ensure completeness of understanding for nontechnical personnel involved in system development and utilization. Since users may relate better to narrative descriptions of software in terms of familiar (existing) functions, a functional description is included. This chapter, therefore, concerns itself with the required functional capabilities of the proposed AMS.

The functional description complements and expands upon the information description. It serves as a vehicle for mutual understanding between the development group and the users of a proposed automated data system [Ref. 29]. Accordingly, it is written in nontechnical language wherever possible. Moreover, it is a dynamic document and further serves as the basis for the test and evaluation master plan discussed in the following chapter.

This chapter outlines the required functional capabilities and presents them in a three section format. First, the functions themselves are proposed along with a processing scheme for implementation. This essentially concerns the installation processing requirements of the end-user. Secondly, external interface requirements are

addressed to include off-ship reporting. Finally, design and security constraints are included to reflect the special processing environment of the application software and potential vulnerability of the data.

## A. FUNCTIONS AND PROCESSING SCHEME

Experience gained through the operation of standard nontactical ADP systems afloat has provided valuable insight into the unique problems and requirements demanded by the shipboard environment. These lessons learned apply to hardware as well as software requirements. This experience has resulted in the selection of an on-line, interactive dialogue interface in both the SNAP configurations. For compatibility reasons, the SNAP design has been selected as the user interface for the proposed AMS.

A principal feature of this design is menu driven software. This facility is presently incorporated in the AN/UYK-62(V) (SNAP II) and certain real-time applications of the AN/UYK-65(V) (SNAP I). The most obvious benefit derived from this approach has been the simplification of system operation for unsophisticated users. In addition, reduced training time requirements have been noted. From a technical standpoint, the proposed menu approach will permit added security in that specific users would only be permitted to view and access functions and data based on their programmed access rights. Restricted capabilities would not even appear on the menu. Moreover, menu implementation serves as a high level driver for software modules thereby enhancing program and data integrity. Finally, menu driven software can support either file or data base management processing as depicted in Chapter 2, Figures 2.2 and 2.3.

Appendices D and E describe the recommended menu structure and formats respectively. The processing scheme represented closely follows the data flow diagrams presented earlier with the menus designed to satisfy 90 to 95 percent of anticipated user requirements. In addition, an "ad hoc" query capability is shown which allows for nonroutine queries and reports. This is implemented by enabling the user to exit to the database environment whereby the data base can be manipulated using the relational algebra of the data manipulation language. A file processing system would require a server program to implement this capability. The target user group for this capability would be work center supervisors, division officers and department heads. These personnel would require additional training in the database/file server program language that is beyond the basic user level.

Specific functional capabilities of the AMS software are formally outlined in the detailed functional description. This document satisfies the provisions of [Ref. 29] and is included in Appendix F. In general, these functions are developed around a transaction processing system with additional provisions to provide management level information.

## B. EXTERNAL INTERFACES

The proposed software must include provisions for interfacing with external activities. For this reason, the automated products should duplicate their manual counterparts in format and content. The objective is to integrate the AMS into the established ammunition management structure and not create the need for a separate support organization for this purpose.

The principal established vehicle for this interface is electronic media, although the processing of manual supply documents is also discussed. A major design consideration which is not addressed in this section is the unique requirements mandated by the end-user's Fleet Commander. Both Atlantic and Pacific Fleet Commanders have promulgated refinements to the basic ammunition management guidance, primarily in the areas of message addressing and deployment operating procedures. For this reason, only generic external reports, established by [Ref. 2], are included in the detailed functional description. Fleet Commander requirements should be addressed during the detailed design step of the software development phase. The appropriate references for this purpose are [Ref. 28] (Atlantic Fleet) and the equivalent instruction promulgated by Commander, Naval Logistics Command, U.S. Pacific Fleet.

## C. DESIGN CONSTRAINTS AND SECURITY

Software development does not occur in a vacuum. Ideally, mutually agreed upon constraints between user and development personnel are available prior to actual development. These constraints may take different forms and originate from various sources. According to Boehm:

> In software engineering, constraints may be self-imposed, as with...availability and performance constraints...or they may be imposed by other conditions, particularly equipment and user limitations or external interface conditions. [Ref. 16]

Whereas the functional description defines the user environment, constraints define the software development environment. Constraints normally take the form of goals and relate to such factors as cost and responsiveness [Ref. 16], maintainability, reliability and human factors [Ref. 30].

The cost constraint is refined during the development phase. It is during this phase that detailed design and implementation issues are addressed. This includes such variables as hardware configuration, target language and implementation strategy (integrated or stand-alone, level of support, etc.). The remaining constraint factors are the subject of the next chapter.

Data security is an overriding goal considering the classification of the data to be processed (Confidential). The appropriate guidance covering classified data processing by naval activities is contained in [Ref. 31]. Fundamental security concerns are discussed in the detailed functional description. However, these are general in nature and should be revised to reflect such considerations as hardware certification level (unclassified or TEMPEST), user environment (single or multi-user), peripheral placement and type of access controls implemented.

# IV. VALIDATION CRITERIA

This chapter outlines the validation testing requirements for the AMS software. Specifically, it defines functional and performance test criteria to be included in the development of a Test and Evaluation Master Plan (TEMP). The TEMP, prepared in accordance with [Ref. 32] and [Ref. 33], is the single most important document of any system's acquisition life cycle. The TEMP serves as a controlling management document which integrates the many development and operational test and evaluation efforts into a single program structure. Moreover, it is the primary document under which acquisition category (ACAT) III and IV programs are managed [Ref. 34]. Due to its nontactical nature and anticipated low dollar threshold, the AMS may be classified as ACAT IV. Accordingly, the importance of this document is significant.

The model for this effort is the SNAP II. There are two major reasons for this. First, the SNAP II presents an interesting study in the test and evaluation for a major ADP system acquisition, many aspects of which parallel the AMS. It is a new, vice replacement, system. The hardware components are commercial off-the-shelf equipment "ruggedized" for the shipboard environment with the addition of power regulators, cabinet air filters, shock mounting and internal modifications to the CPU rack. Software design and system support philosophy are also complementary with that proposed in the Detailed Functional Description. Secondly, by addressing early-on the design goal of compatibility with SNAP II in the validation criteria, a more viable software product can be developed.

The thrust of this chapter is to develop software validation test criteria which address the desired performance characteristics of the final product. Functional performance characteristics are provided in Appendix F.

## A. PERFORMANCE BOUNDS

Reliability, maintainability and availability (RMA) are the primary measurement areas of system performance. These design parameters are inherent characteristics of system or product design [Ref. 30] and have minimum thresholds established for them. For our purpose, these thresholds are identical to the SNAP II operational evaluation test criteria established in [Ref. 35].

The first design parameter, reliability, is defined by Blanchard [Ref. 30:p. 23] as the probability that a system or product will perform in a satisfactory manner for a given period of time when used under specified operating conditions. He further provides a mathematical function relating reliability to time:

$$R(t) = 1 - F(t) = \int_t^{\infty} f(t)\, dt$$

In the above, F(t) represents the probability of failure by time t, and f(t) is the density function of the random variable t. Assuming that time to failure is described by an exponential density function, Blanchard replaces f(t) with:

$$e^{-t/\Theta} / \Theta$$

where $\Theta$ is the mean time to failure. When this is integrated for time t it yields:

$$R(t) = e^{-t/\Theta}$$

Finally, since the Mean Time Between Failure (MTBF = $\Theta$) is equal to $1/\lambda$, where $\lambda$ is the failure rate, R(t) may be rewritten as:

$$R(t) = e^{-t/MTBF} = e^{-\lambda t}$$

The failure rate can be obtained by dividing the number of failures by the total operating hours. The MTBF for a system is the reciprocal of this formula. As an example, if a system experienced two failures in 352 hours of mission time, the MTBF would be 176 hours computed as follows:

$$\lambda = 2/352 = 0.0056818$$

$$MTBF = 1/\lambda = 1/0.0056818 = 176 \text{ hours}$$

A "critical failure" is defined as a casualty to the software that reduces operability by fifty percent at the system level; a "major failure" is defined as a loss of an important capability but the system has at least fifty percent capacity remaining [Ref. 35]. These definitions will be used in calculating the MTBF for the AMS. A minimum MTBF threshold of 2000 hours is established per [Ref. 35].

Maintainability is the second performance characteristic and, like reliability, is a design parameter. Unlike reliability, however, there is no single metric for measurement purposes. This is due to the requirement to consider manifold factors in its assessment. Blanchard lists sixteen such factors including Mean Time Between Maintenance (MTBM), Mean Time Between Replacement (MTBR) and Mean Time To Repair (MTTR) [Ref. 30:p. 15]. Since maintainability relates to a system's ability to be maintained, the maintenance philosophy employed will have a major impact on the selection and application of the metrics to be used. As an example, a system which is designed to give the ship's force full diagnostic capability for software casualties would tend to stress the Mean Time To Fault Locate (MTFL) measure over other measures.

Maintainability, therefore, may be defined as a combination of factors such as:

1. A characteristic of design and installation which is expressed as the probability that an item will be retained in or restored to a specified condition within a given period, when maintenance is performed in accordance with prescribed procedures and resources.
2. A characteristic of design and installation which is expressed as the probability that maintenance will not be required more than x times in a given period, when the system is operated in accordance with prescribed procedures. This may be analogous to reliability when the latter deals with the overall frequency of maintenance.
3. A characteristic of design and installation which is expressed as the probability that the maintenance cost for a system will not exceed y dollars per designated period of time, when the system is operated and maintained inaccordance with prescribed procedures. [Ref. 30:p. 15]

The metrics to be used in determining maintainability for the proposed AMS are MTFL, mentioned earlier, and the Geometric Mean Time To Repair ($MTTR_g$). The MTFL is computed by averaging the times to locate a fault (actual or inserted for test purposes). The $MTTR_g$ is the geometric mean of the

distribution of times to repair for critical and major failures and is computed as:

$$MTTR_g = \log^{-1} [ \Sigma(\log t_i) / N ], \quad i = 1 \text{ to } N$$

where $t_i$ is the time to repair the ith failure and N is the number of critical and major failures. Maximum thresholds for these metrics are established as 90 minutes for $MTTR_g$ and 45 minutes for MTFL.

The final performance characteristic to be evaluated is availability. The applicable metric for this purpose is Operational Availability ($A_0$) and is defined as the probability that a weapon system will be in an operable state at a random point in time (a measure of functional readiness) [Ref. 36]. It is calculated by using the following formula:

$$A_0 = \text{uptime} / (\text{uptime} + \text{downtime})$$

Policy guidance for the application of $A_0$ is provided in [Ref. 37]. In addition, this instruction mandates the use of $A_0$ as the governing measurement in determining a system's overall worth to the Navy. The threshold to be used as the criterion for the AMS is an $A_0$ greater than or equal to 85 percent. The $A_0$ metric should be determined and evaluated for both software and hardware separately, at the system and component level as appropriate. Refinements to the basic $A_0$ formula should also be made to accurately reflect the logistics factors necessitated by the maintenance and support philosophy selected. This should

include Mean Supply Response Time (MSRT) and component maintenance queue time.

## B. CLASSES OF TESTS

The structural design of the AMS software permits a structural approach to the testing (and subsequent maintenance effort) of the software product. Testing, per se, is not a separate phase of the software life cycle and often goes on in parallel with programming [Ref. 38]. Moreover, the validation effort includes the continuous review of such concerns as documentation and data requirements in addition to the testing of source code. The constellation of activities under the umbrella term "validation testing" therefore, are allocated to test phases vice life cycle phases.

There are two principal classes of Navy test and evaluation phases. These are the development test and evaluation and the operational test and evaluation [Ref. 33:p. 2]. Each test phase is further divided into subphases along a project's life cycle and emphasizes the testing of characteristics associated with that life cycle phase. As an example, Development Test and Evaluation I (DT-I) is conducted during the demonstration and validation (life cycle) phase and is designed to demonstrate those areas of concern to be reviewed at Milestone II. Due to the modular (structured) architecture of the software, however, more than one test phase may be occuring during a given life cycle phase. As a result, operational testing may be conducted on the Initial Operating Capability (IOC) at the same time development testing is being performed on a new communication interface module in Release 1.

The purpose of this section is to discuss major development and operational testing considerations without

regard to a particular life cycle definition. The first test phase, development testing, is divided into the three subphases of unit, integration and benchmark testing. Operational testing is viewed as a follow-on test phase to development testing and, for our purpose, is not further subdivided.

1. Development Testing

The purpose of development testing is to assist the engineering design and development process and to verify the attainment of technical performance specifications and objectives [Ref. 33:p. 2]. A major requirement of this test phase is that it be conducted in a controlled environment. This is necessary to reduce the variables inherent in computer performance analysis by explicitly defining the test jobs and the environment in which these tests are executed [Ref. 39]. Development testing serves an additional function besides assisting the engineering design and development process. It provides a means to refine test criteria and develop baseline hardware and software test configurations.

The first subphase of development testing is unit testing. This phase, according to Pressman, focuses verification effort on the smallest unit of software design: the module. He further lists five module characteristics to be evaluated during this effort [Ref. 7:p. 296]:

a. The module interface.
b. Local data structure.
c. Important execution paths.
d. Error handling paths.
e. Boundary conditions affecting all of the above.

Unit testing is process ("white box") oriented. The module is provided with test data and is "driven" by a program

developed for that purpose. "Stub" modules are developed to test interface capability. This effort should concentrate on the integrity of data flow across the interface and include selective testing of execution paths. Finally, user involvement should be provided at appropriate points by the scheduling of design reviews and formal walk throughs for each module.

Integration testing follows the unit testing subphase and is defined as follows:

> Integration testing is a systematic technique for assembling software while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a software structure that has been dictated by design [Ref. 7:p. 299].

Integration testing may be conducted "top-down" wherein modules are incorporated by moving down through the control hierarchy, or "bottom-up" wherein assembly and testing begins with the lowest-level modules and proceeds upward in the control hierarchy. Each approach has inherent benefits and problems to be considered. A major strength of top-down integration is the verification of major control or decision points early-on in the test process. A drawback, however, is the need for stubs which create added overhead, and the attendant testing difficulties associated with them. Bottom-up integration eliminates the need for stubs as subordinate modules to a given level are always available prior to testing. The negative aspect of this approach is that the complete program as an entity is not realized until the last module is added [Ref. 7:pp. 300-302]. This fact may hinder conceptual management of the testing process by both user and development personnel.

With regard to the divergent approaches described above, Pressman provides the following recommendation in selecting an integration strategy:

> Selection of an integration strategy depends on software characteristics and sometimes, project schedule. In general, a combined approach that uses the top-down approach for upper levels of the software structure, coupled with a bottom-up approach for subordinate levels, may be the best compromise. [Ref. 7:p. 302]

Benchmark testing is the third and final subphase of development testing. It follows integration testing and involves the demonstration of a baseline software configuration on a vendor-provided hardware suite using a "typical workload." The workload "package" consists of software programs and data files and is prepared by the development authority. This serves as the benchmark for test purposes.

The Federal benchmark process may be broken down into six phases [Ref. 40]. These are workload definition and analysis, design and construction, testing, agency package preparation, vendor preparation, and demonstration. Success of the benchmark effort is highly dependent on accurate definition and construction of the workload package. The GAO has noted that poor design and documentation of the benchmark is a major cause of frustration in the acquisition process and results in additional cost burdens to both the agency and the vendor.

There is an inherently high cost associated with benchmark testing even with a properly designed package. For this reason, the GAO recommends the limiting of benchmarking to those ADP projects where total acquisition cost justifies the additional cost and burden. Therefore, the applicability of this test to the AMS program should be carefully determined during the development phase when project scope and implementation strategy are refined. It is included here, however, as an option for consideration. Since benchmarking provides the first (albeit costly) opportunity to evaluate the software outside of the

cleanroom environment, the benefits to be derived include the refinement of test and software documentation, coding, user interface and support and maintenance philosophies.

2. Operational Testing

A principal distinguishing characteristic of operational testing is the absence of a controlled environment. The objectives of this phase are to estimate a system's operational effectiveness and operational suitability, identify the need for modifications and provide information on tactics [Ref. 33:p. 3]. Accordingly, testing is conducted in the actual operating environment using production hardware and fleet issue software. In addition, system operation and maintenance are performed by personnel from the user population under realistic conditions.

It is during operational testing that the RMA metrics are applied and evaluated. In addition, other evaluation criteria relating to operational suitability and effectiveness should be applied. These are discussed in Section D, Special Considerations.

C. EXPECTED SOFTWARE RESPONSE

The quality of a software product, as judged by the user, hinges on the responsiveness of the software to his needs. Accuracy of functional design with regard to the user's imperatives is on determinant in the ultimate success in user acceptance. Another major determinant in this area is the quality of the interface. Keen and Scott Morton [Ref. 41] note that the system, as seen by the users, is the interface. The "interface," as used here, primarily concerns the menus and on-line display presentations to the user. However, the interface encompasses more attributes, the importance of which are not so obvious to successful

user acceptance. These include such design issues as timing, communicability, robustness (ability to recover and reliability), and ease of control. Recent studies have even alluded to the need for congruence between the system operation scheme and the user's cognitive processes [Ref. 41].

While the more tangible criteria of user acceptance are provided in the next section and in the Detailed Functional Description, a need exists to consider those "gray areas" which are, nevertheless, important in the final acceptance of the product. One proposal to do this is to include the user in the design process of the software. The degree of involvement should be determined on the technology level and phase of the product life cycle. The reader is referred to [Ref. 42] for an interesting discussion on this strategy.

## D. SPECIAL CONSIDERATIONS

This section concerns the subjective validation criteria relating to operational suitability and effectiveness of the proposed software. The treatment of these criteria is necessarily cursory as they do not have established thresholds and depend, to a large part, on judgement in their use and evaluation. Their importance lies in their ability to provide a more complete picture of a system's effectiveness.

Under the first category of operational suitability there are six evaluation factors. These include logistic supportability, compatibility, interoperability, training requirements, human factors and safety. The objective is to assess the adequacy of the maintenance and support philosophies employed. Specific documentation to be used in support of this evaluation include the Integrated Logistic Support Plan (ILSP), Provisioning Allowance Parts List

(PAPL), Navy Training Plan (NTP), the Detailed Functional Description and field level documentation such as user's manuals and technical manuals. The test scenarios will concern the ability of users to operate and maintain the system within the established support structure and not stress discrete tasks as in the case of RMA.

The second category of operational suitability evaluates a system's ability to achieve design objectives within established constraints. This area is divided into user effectiveness and unattended operations.

User effectiveness is evaluated by observing the system operation and determining the corresponding productivity of the user. This is done by estimating manhour requirements for manual and automated modes of performing the same task and comparing the relative times. Other criteria include accuracy of reports prepared and data maintained under both methods. Unattended operations is another characteristic of operational effectiveness and is a function of the hardware and software configurations and processing scheme. It is a metric calculated as the percentage of the hours of unattended operations ($\delta$) as follows:

$$\delta = [\ UO\ /\ (UO + AO\ )\ ]\ x\ 100$$

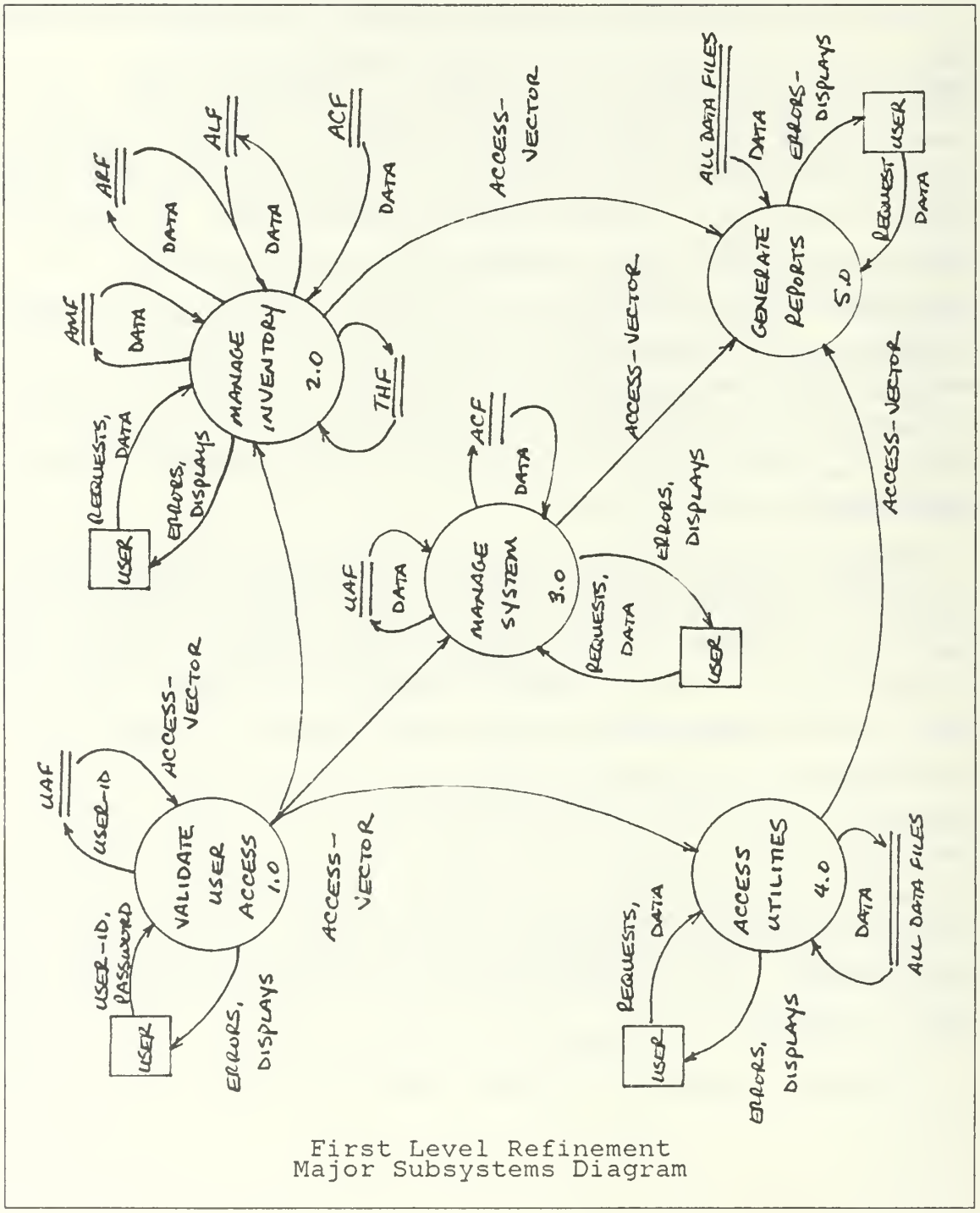
where AO is the number of hours of attended operations and UO is the number of unattended operations. The threshold for this metric may be a statistical range based on a combination of hardware vendor specifications and software process observations obtained during development testing.

# V. CONCLUSION

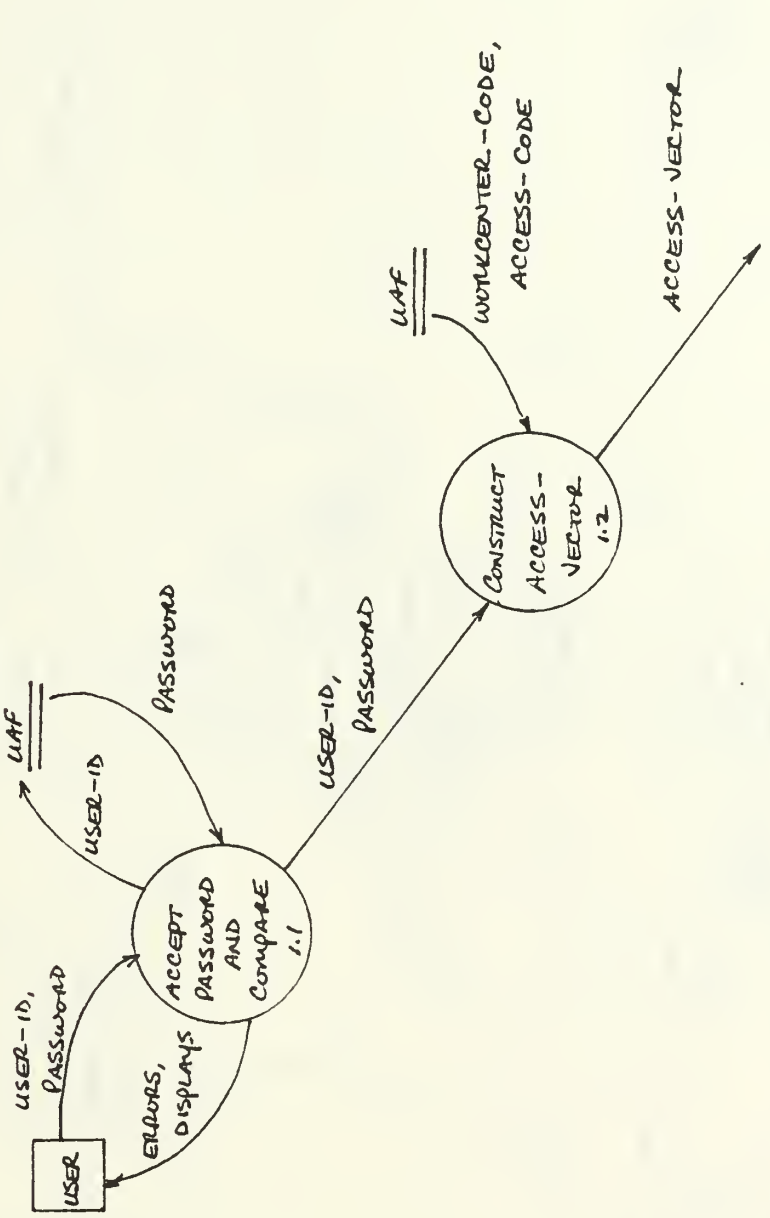
The automated application proposed in this thesis is one feasible alternative to the present manual methods associated with ammunition inventory management and reporting at the afloat end-user level. A logical design for this management and reporting system has been developed within the overall goals of SNAP compatibility, increasing user productivity, and improving the end-user/CAIMS interface. This effort has resulted in an initial software design specification that is independent of hardware configuration and programming language. As such, it serves as a foundation for subsequent development efforts and for tailoring the application program to unique Fleet and user requirements.

Future research should further refine and develop the design for this software, with the additional possibility of tailoring the AMS for shore-based applications. In the latter effort, a design goal of compatibility with the Base and Station Information System (BASIS) program should be incorporated for reasons similar to those for seeking SNAP compatibility for the onboard version of the AMS. Future development on the basic AMS design should address changes in ammunition management procedures, technological changes and new requirements demanded by the user and the afloat environment. In this way the currency and viability of the AMS is ensured.

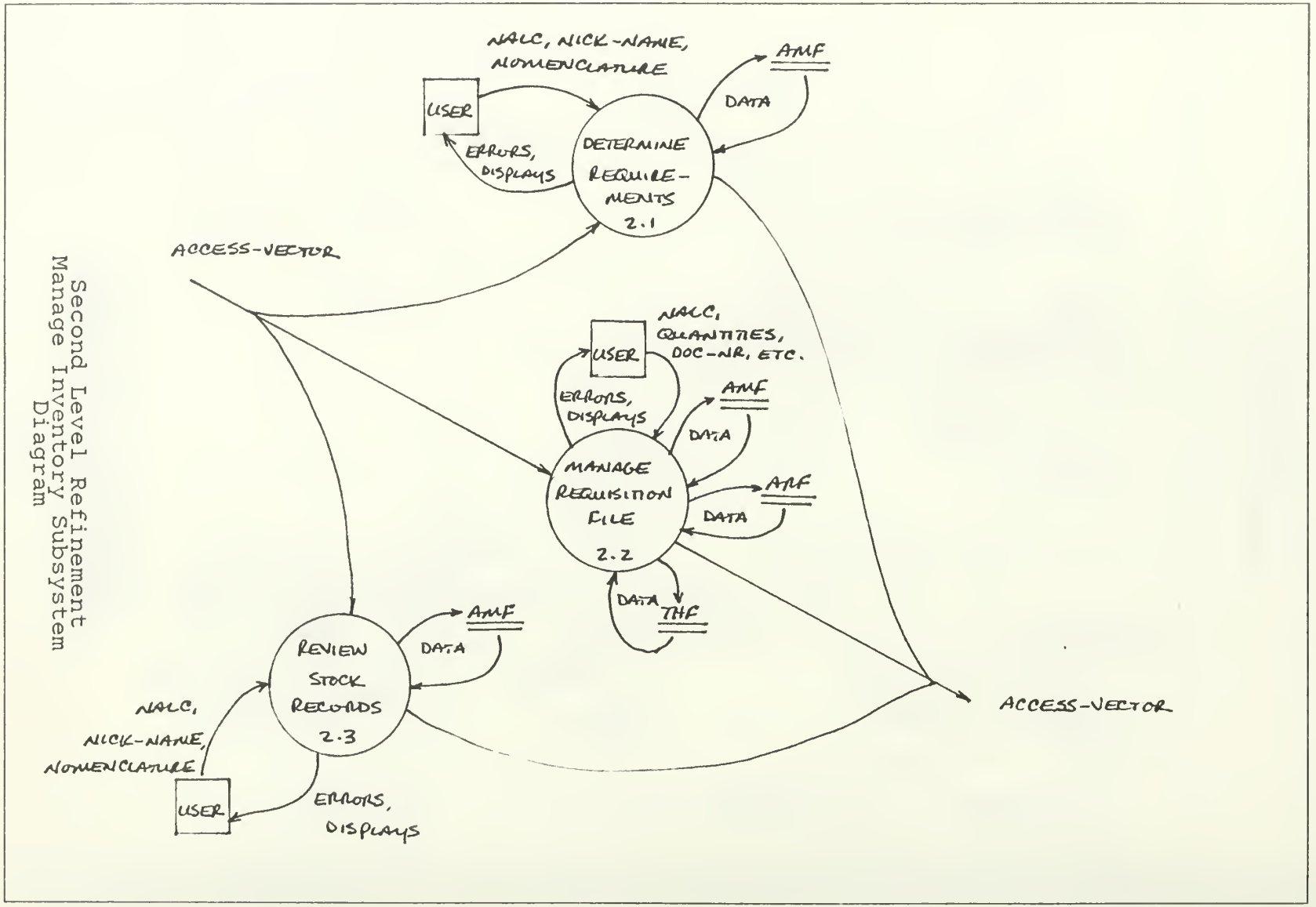
APPENDIX A
Data Flow Diagrams



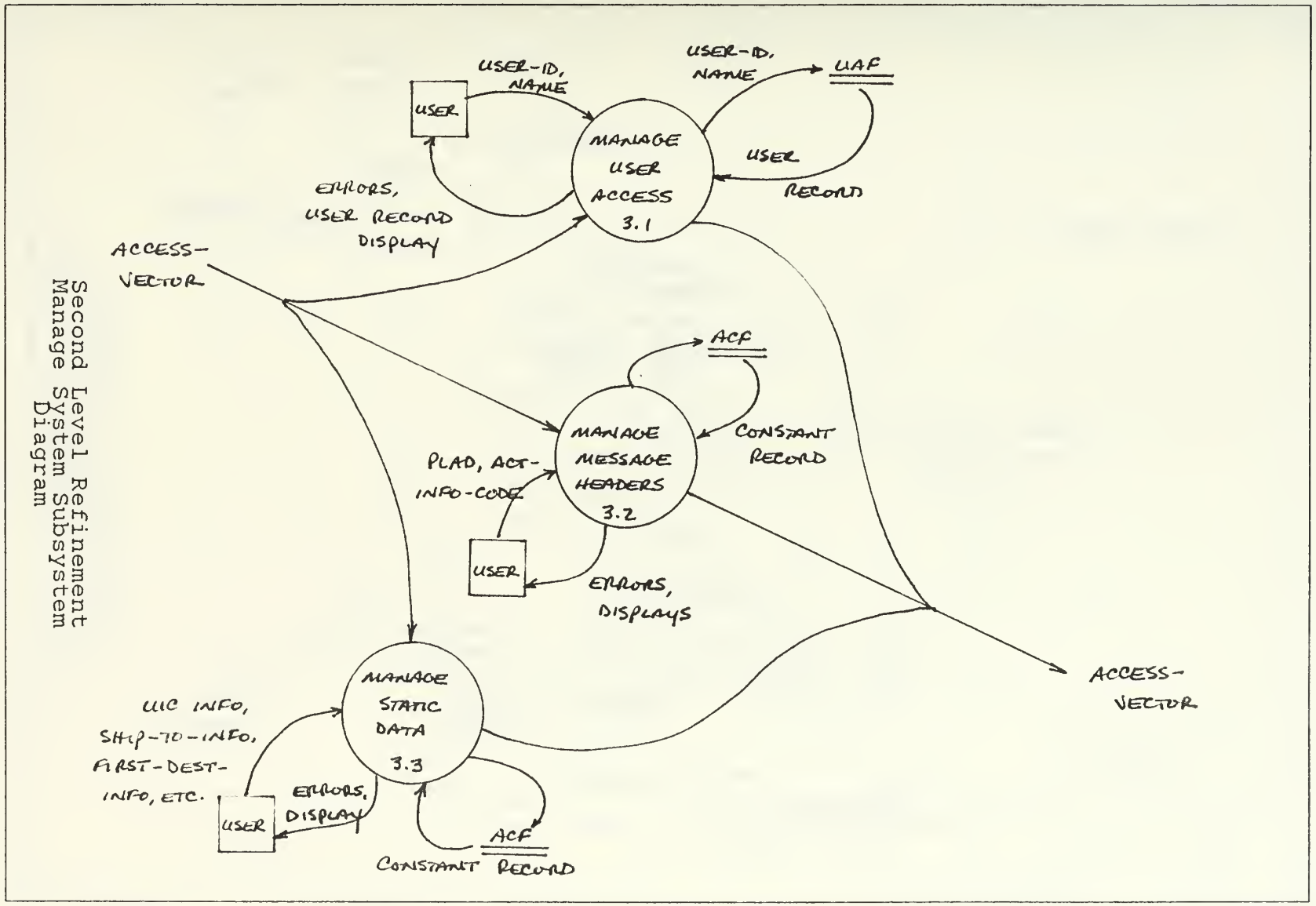

First Level Refinement
Major Subsystems Diagram



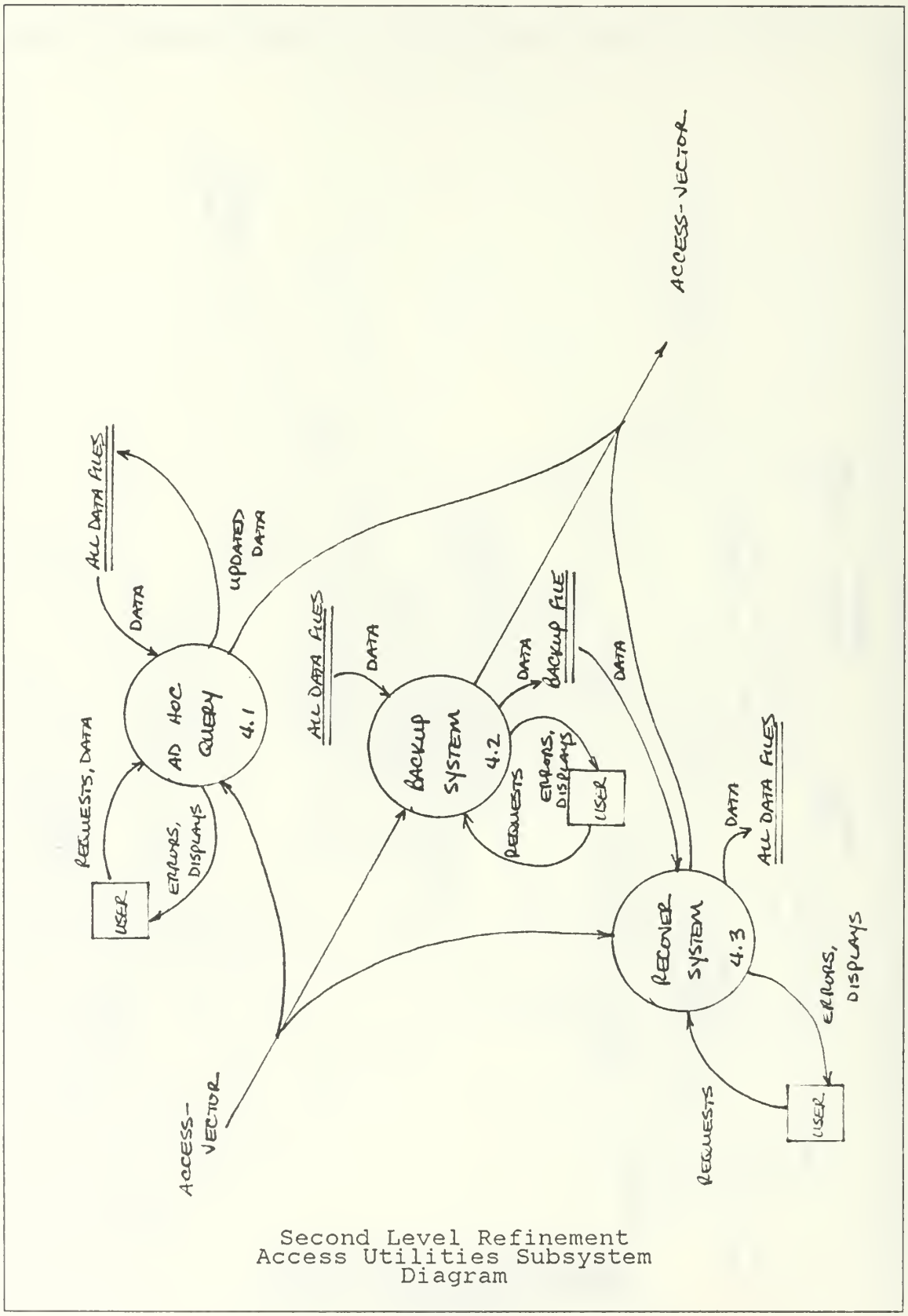

Second Level Refinement
Validate Access Subsystem
Diagram



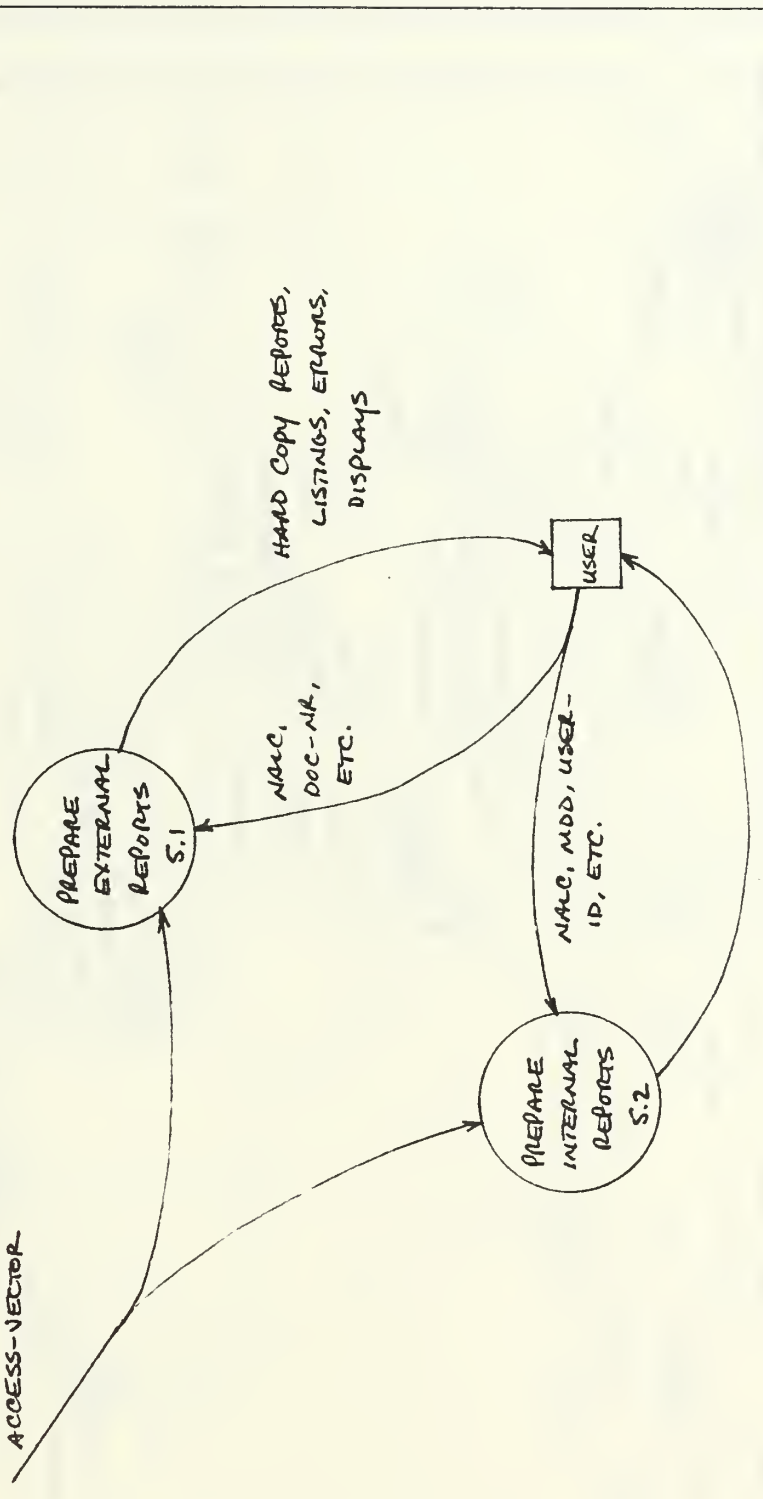

Second Level Refinement
Manage Inventory Subsystem
Diagram



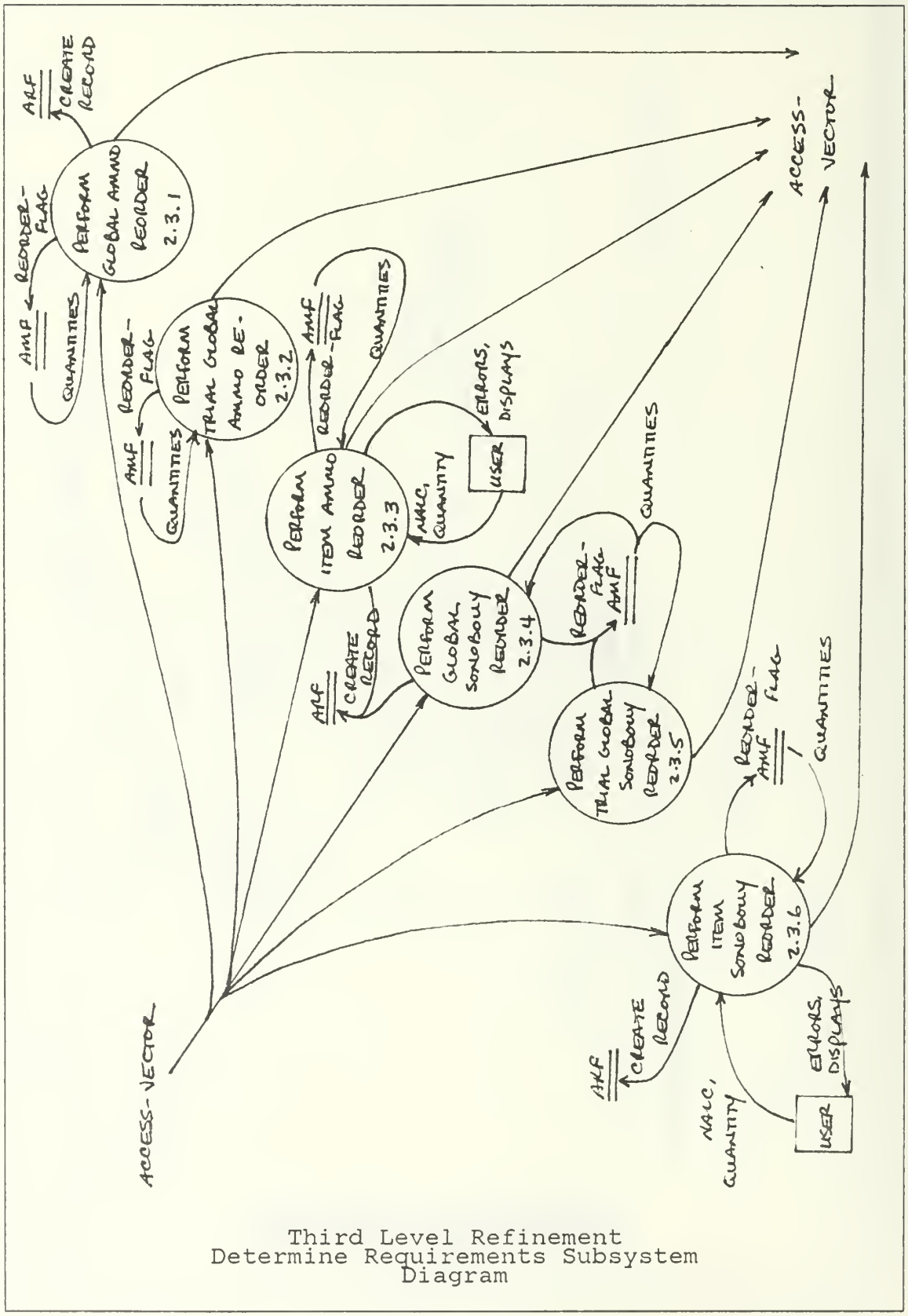

Second Level Refinement
Manage System Subsystem
Diagram



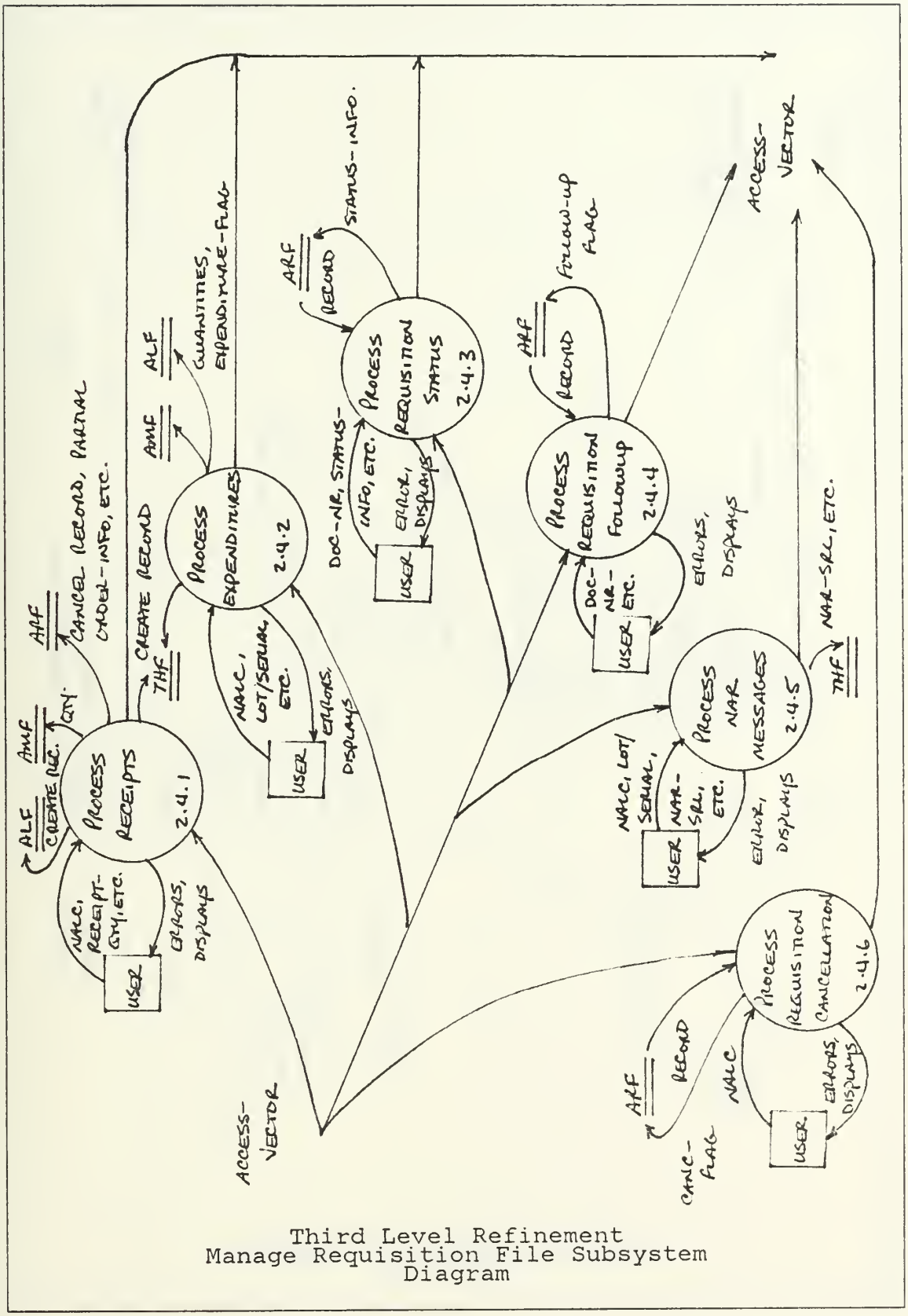

Second Level Refinement
Access Utilities Subsystem
Diagram



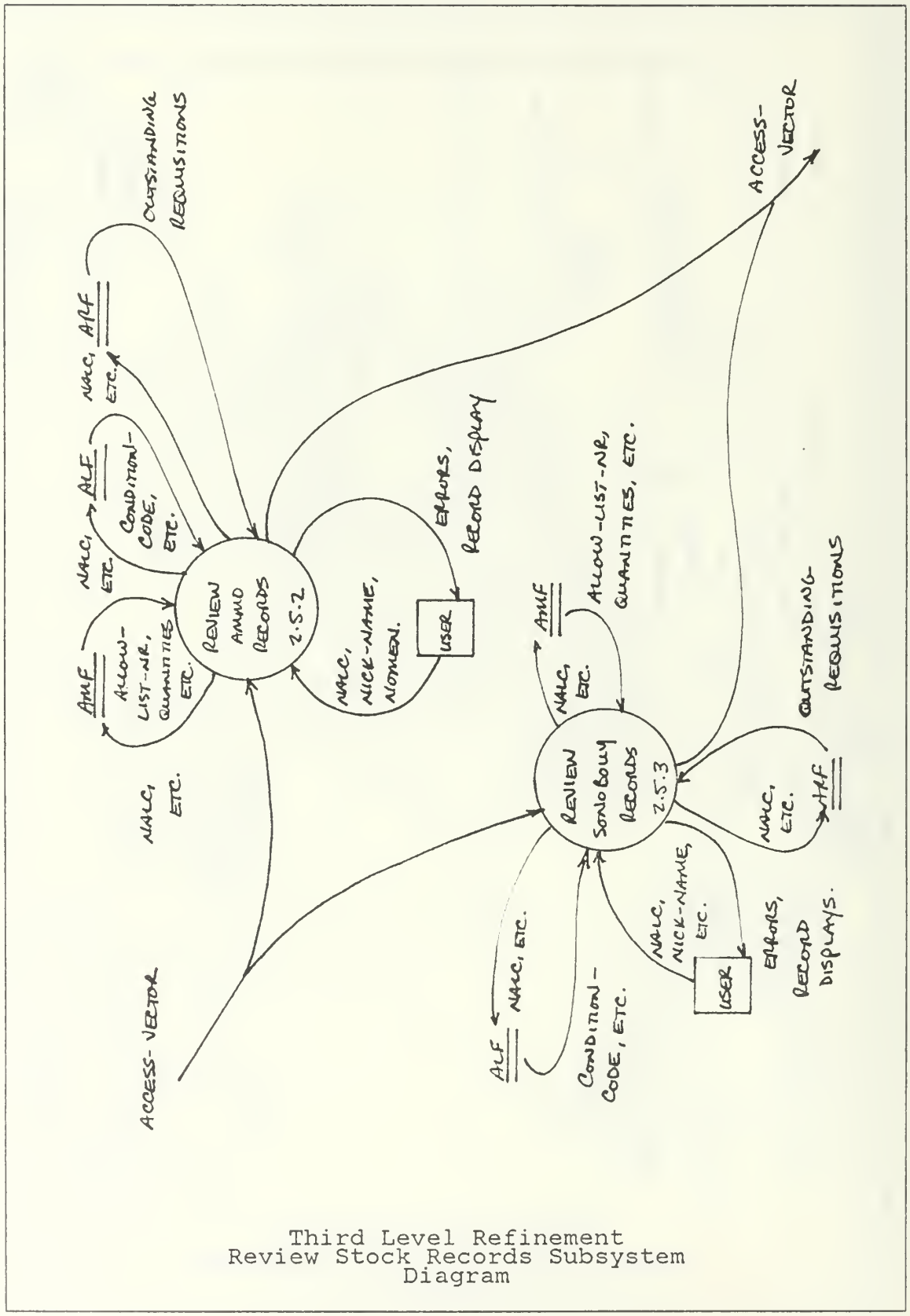

Second Level Refinement
Generate Reports Subsystem
Diagram



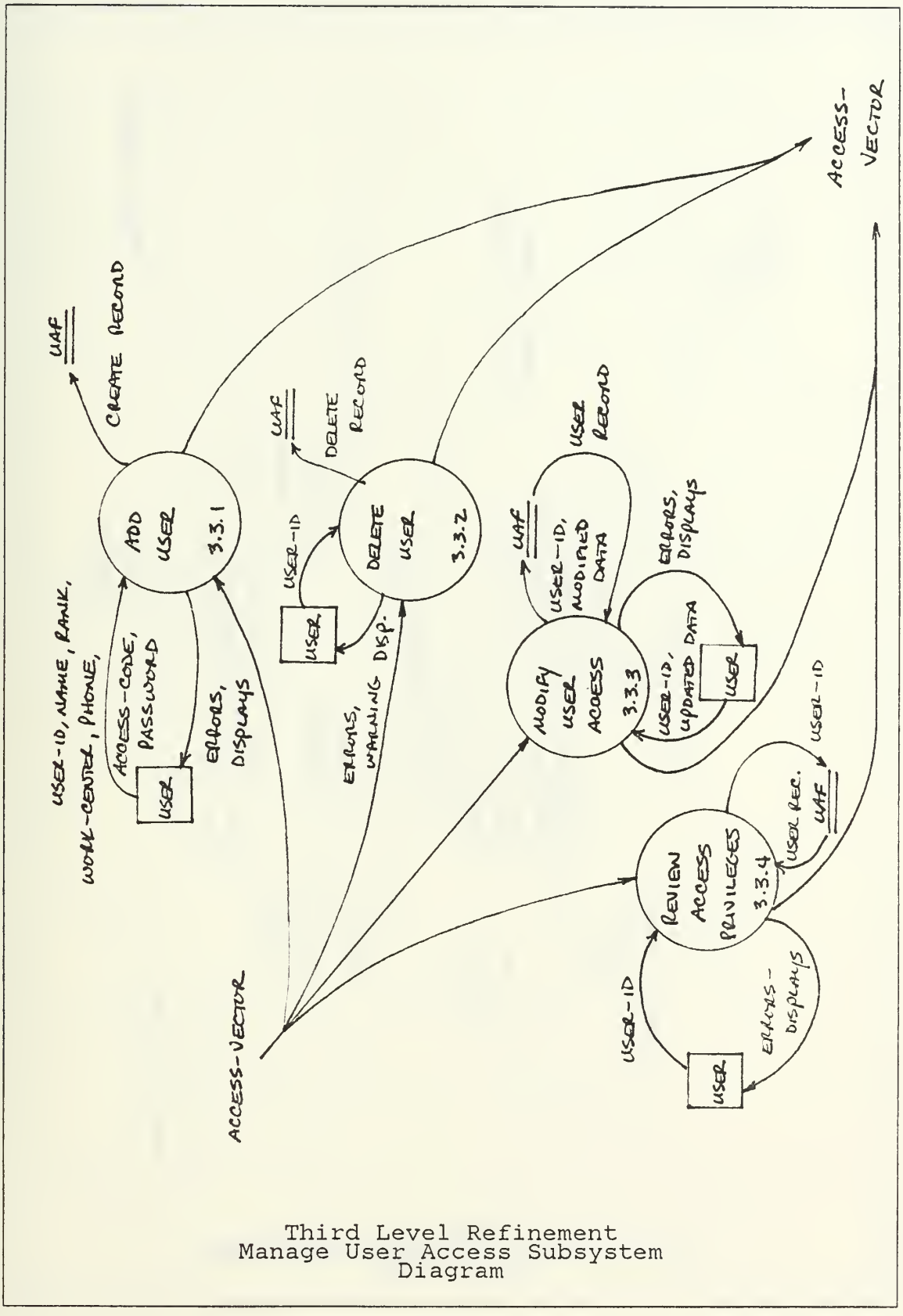

Third Level Refinement
Determine Requirements Subsystem
Diagram



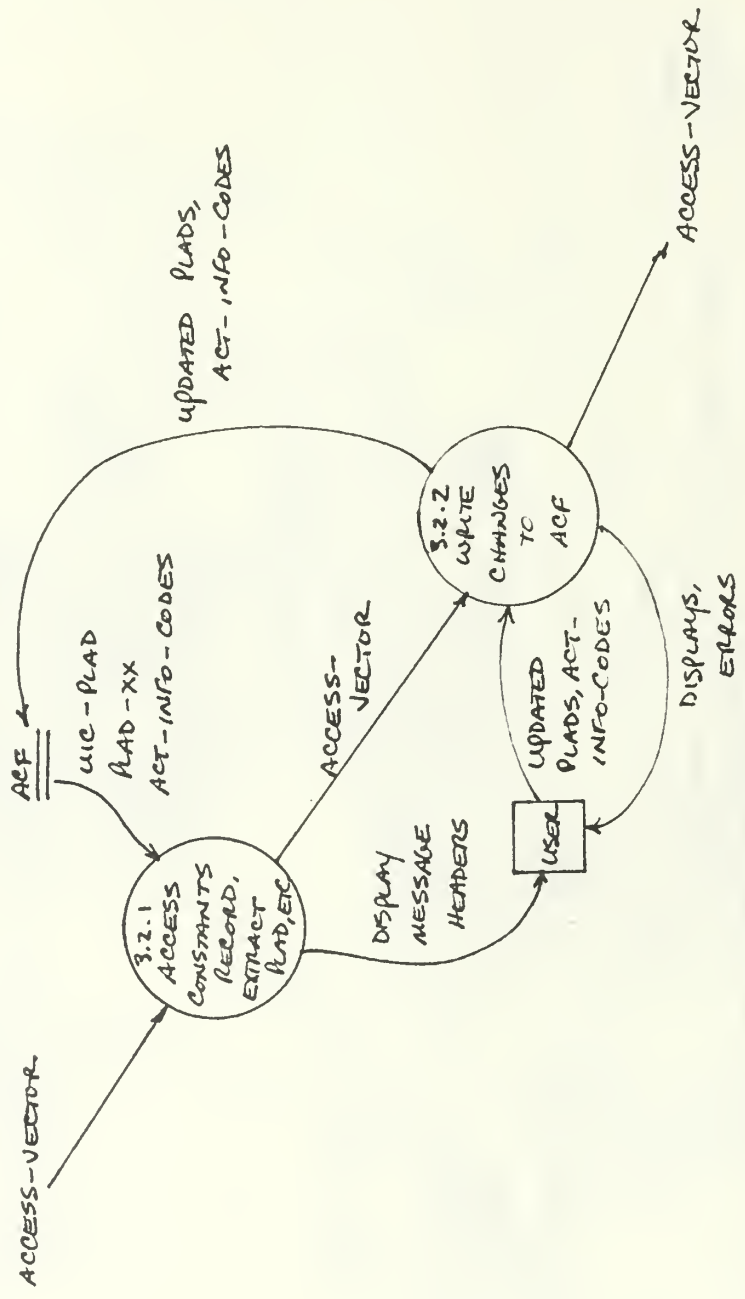

Third Level Refinement
Manage Requisition File Subsystem
Diagram



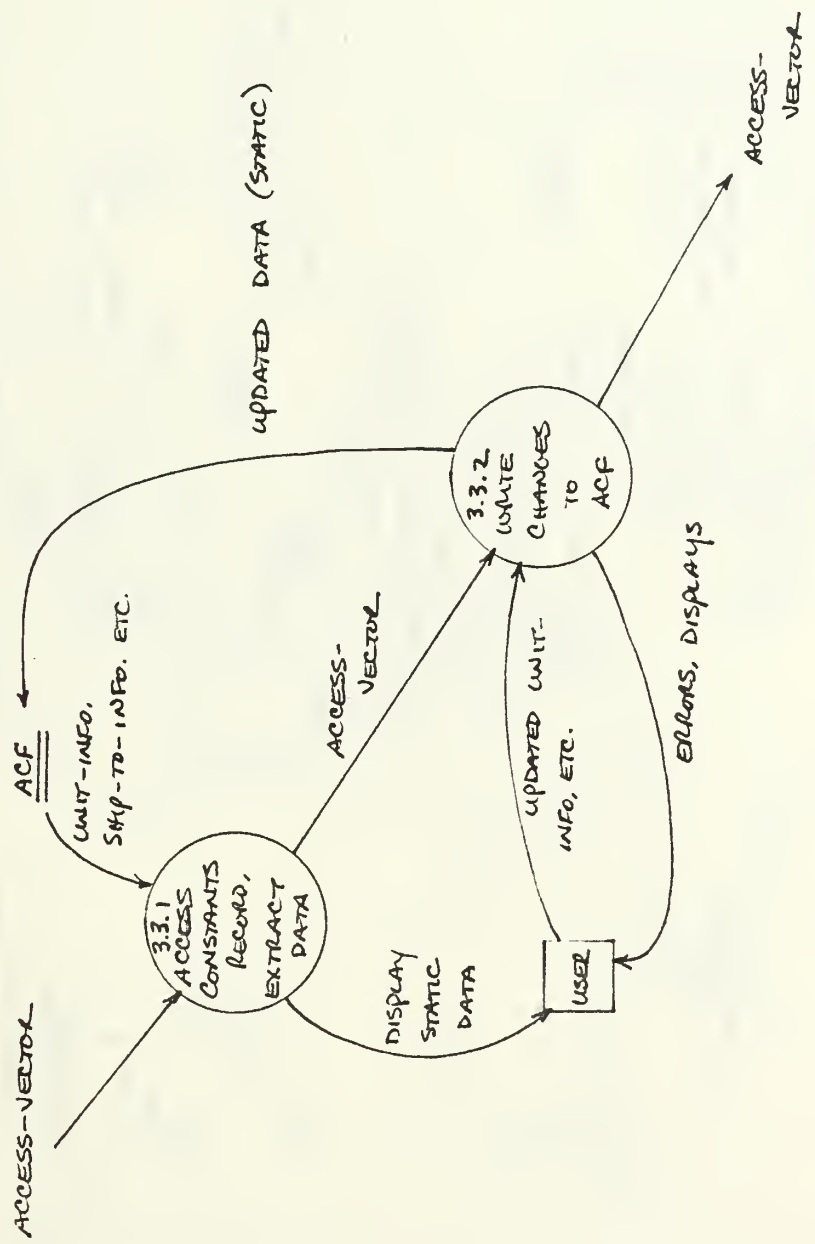

Third Level Refinement
Review Stock Records Subsystem
Diagram



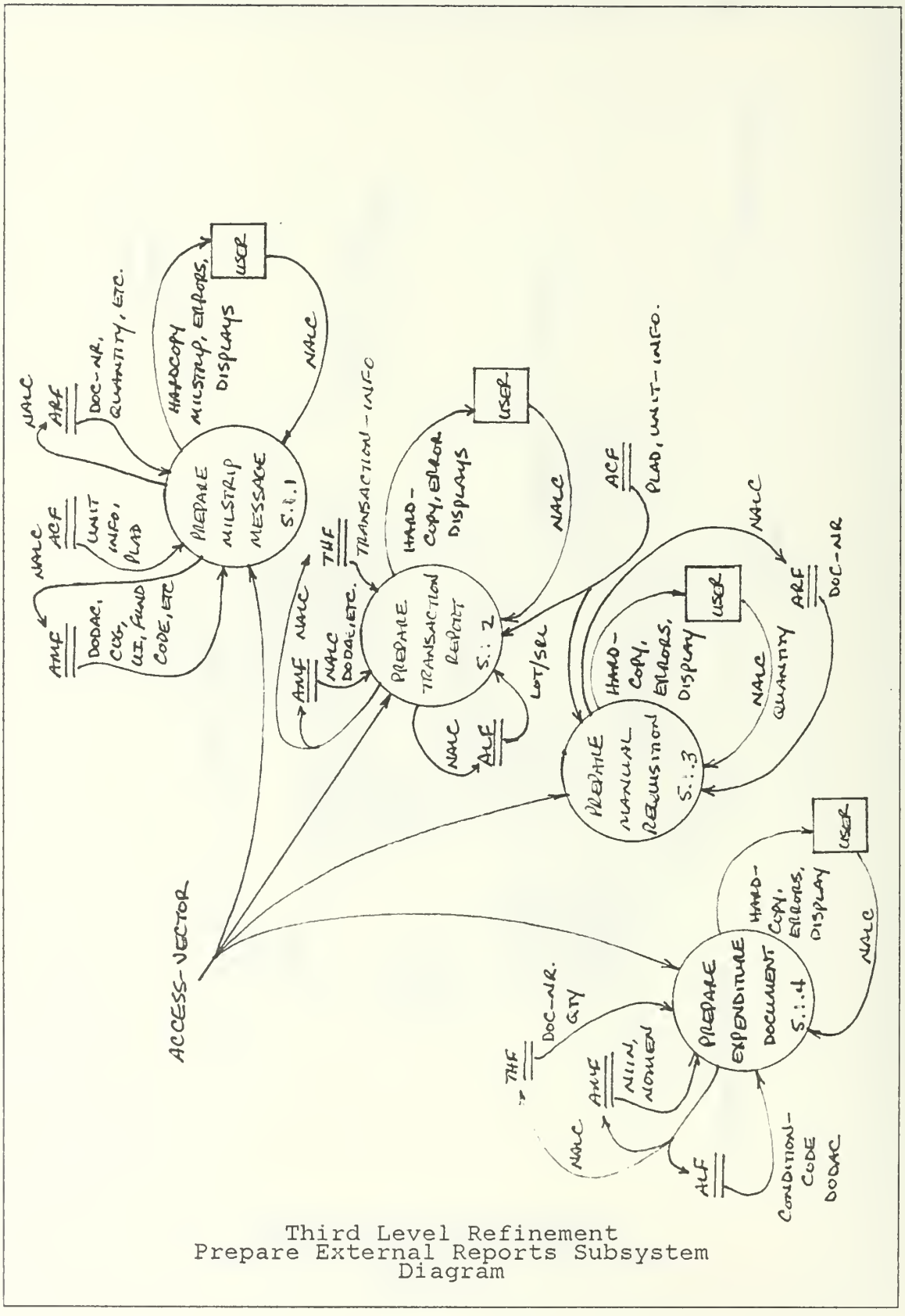

Third Level Refinement
Manage User Access Subsystem
Diagram




Third Level Refinement
Manage Message Headers Subsystem
Diagram




Third Level Refinement
Manage Static Data Subsystem
Diagram




Third Level Refinement
Prepare External Reports Subsystem
Diagram

Third Level Refinement
Prepare Internal Reports Subsystem
Diagram

APPENDIX B
Data Dictionary

DEN:
TITLE: Access Code
COBOL: ACCESS-CODE
PIC: 9
DESC: A numeric code indicating user access privileges.
NOTES: A subfield of ACCESS-VECTOR.
CODES: 0 System Manager.
1 Ad Hoc Query capability, record creation, update and deletion capability.
2 Record creation, update and deletion capability.
3 Record update capability only.
4 Record read capability only.
9 No record access.

---

DEN:
TITLE: Access Vector
COBOL: ACCESS-VECTOR
PIC: No PIC
DESC: Identifies the system user and the access privil-edges/capabilities assigned.
NOTES: Constructed by a grouping of 4 subfields: USER-ID, PASSWORD, WORK-CENTER-CODE and ACCESS-CODE.

---

DEN: A001
TITLE: Activity Routing Identifier
COBOL: ACTIVITY-ROUTING-IDENTIFIER
PIC: AXX
DESC: A three-digit alphabetic or alphanumeric code assigned to Inventory Control Points, Inventory Managers, distribution points and designated stor-age points. The routing identifier utilized on a document or transaction serves to indicate one of the following: The intended recipient of the document; The actual shipper or consignor (See the SDED, DEN A001 for further information).
NOTES: Used in CAIMS as PIC XXX.
REFS: NAVSUP Pub 437
DoDI 4140.17M Supplement 1
ORIG: 9621
CREATED: 29 APR 82 UPDATED: 11 APR 84
USER: CAIMS LEVEL II TRIDENT UADPS-SP UICP

---

DEN:
TITLE: Addressee Plain Language Address
COBOL: ADDEE-XX-PLAD
PIC: X(20)
DESC: The plain language communications address for external addressees. Serialized 01 to 06.
NOTES: Serial number replaces "XX." Matched one-for-one with the associated ADDEE-XX-ACT-INFO-CODE.
REFS: NTP 3(F)

---

DEN:

TITLE: Addressee Action/Information Code
COBOL: ADDEE-XX-ACT-INFO-CODE
PIC: 9
DESC: Designates the type of message (ATR, etc.) and distribution type (action/information) for a given PLAD. Serialized 01 to 06.
NOTES: Serial number replaces "XX." Matched one-for-one with the associated ADDEE-XX-PLAD.
CODES:

| | | |
|---|---|---|
| 0 | Action: MILSTRIP,STR,ATR | Info: NONE |
| 1 | Action: MILSTRIP,ATR | Info: STR |
| 2 | Action: MILSTRIP | Info: ATR,STR |
| 3 | Action: ATR,STR | Info: MILSTRIP |
| 4 | Action: ATR | Info: MILSTRIP,STR |
| 5 | Action: STR | Info: MILSTRIP,ATR |
| 7 | Action: NONE | Info: MILSTRIP,ATR,STR |
| 8 | Action: (Future use) | Info: |
| 9 | Action: (Future use) | Info: |

---

DEN: K026A
TITLE: Advice Code
COBOL: ADVICE-CODE
PIC 9A
DESC: Advice codes are numeric/alphabetic and flow from requisition originator to initial processing point. The purpose of advice codes is to provide coded instructions to supply sources when such data are considered essential to supply action and narrative form is not feasible.
NOTES: Used in MISIL, I/O-1, AP/OP-E20, UADPS-E20, UADPS-SP, SUADPS
REFS: MILSBILLS
NAVSUP Pub 437
ORIG: DBA
CREATED: 22 APR 76 UPDATED: 13 APR 84
USER: IDA LEVEL II MISIL UADPS-SP UICP

---

DEN: E372
TITLE: Allowance Quantity
COBOL: ALLOW-QTY
PIC: 9(5)
DESC: The total item quantity computed for allowance requirements during provisioning.
ORIG: DBA
CREATED: 16 FEB 77 UPDATED: 6 FEB 84
USER: CAIMS UICP

---

DEN: E306
TITLE: Ammunition Allowance List Number
COBOL: AMMO-ALOWC-LIST-NR
PIC: 9(5)
DESC: An identification number assigned to an activity allowance list which also denotes the type of allowances contained on the list.
NOTES: Ships Service Lists are numbered from 30,000 to 33,999; Fleet Issue/Cargo Load Lists are numbered from 34,000 to 34,499; and Miscellaneous Lists are numbered from N638,000 to _. Used in CAIMS as PIC X(5).
ORIG: DBA
CREATED: 12 AUG 69 UPDATED: 3 FEB 84
USER: CAIMS

---

DEN: C301
TITLE: Ammunition Lot Number
COBOL: AMMUNITION-LOT-NUMBER
PIC: X(16)
DESC: A number assigned at the time of manufacture or assembly to identify a group of rounds of ammunition, each component of which is manufactured by one manufacturer under uniform conditions, and which are expected to function in a uniform manner. For complete rounds of assembled ordnance (missiles, etc.) this field will contain the item serial number.
NOTES: Used in CAIMS as PIC X(16) in NXN and MHF File.
REFS: A. Lot Data Cards B. TW024-AA-ORD-010; Ammunition-Unserviceable, Suspended and Limited Use.
ORIG: DBA
CREATED: 1 SEP 68 UPDATED: 18 NOV 83
USER: CAIMS

---

DEN:
TITLE: Ammunition Transaction Report Condition Code
COBOL: ATR-CONDITION-CODE
PIC: X
DESC: The condition code for an ATR reported lot/serial number of ordnance.
CODES: Codes are listed in SPCCINST P8010.12.

---

DEN: D219
TITLE: ATR Transaction Code
COBOL: ATR-TRNSN-CODE
PIC: A(1)
DESC: A code which designates the type of transaction in an ATR transaction line.
CODES: Applicable codes are listed in SPCCINST P8010.12.
ORIG: 8511
CREATED: 20 MAR 84 UPDATED:
USER: CAIMS

---

DEN:
TITLE: ATR Transaction Date
COBOL: ATR-TRNSN-DATE
PIC: 9(4)
DESC: The Julian date of the actual transaction of a given lot/serial number of ordnance.

---

DEN:
TITLE: ATR Transaction Flag
COBOL: ATR-TRNSN-FLAG
PIC: X
DESC: A record "flag" field which is set to indicate that a transaction has occured for a given lot/serial number and requires ATR reporting.

---

DEN: D220
TITLE: ATR Transaction Quantity
COBOL: ATR-TRNSN-QTY
PIC: 9(9)
DESC: A number up to nine digits which represent quantity application to a transaction.
ORIG: 8511

CREATED: 20 MAR 84 UPDATED:
USER: CAIMS

---

DEN:
TITLE: Balance Estimated Shipping Date
COBOL: BALANCE-ESD
PIC: 999
DESC: The three digit date (Julian date less leading year digit) of the estimated shipping of the balance of items remaining for a given requisition. This is provided in status documentation by the requisition holding activity.

---

DEN:
TITLE: Balance Quantity
COBOL: BALANCE-QUANTITY
PIC: 9(8)
DESC: The requisition quantity remaining under a requisition number following partial filling by a supply activity. This is provided in status documentation by the requisition holding activity.

---

DEN:
TITLE: Cancellation Flag
COBOL: CANC-FLAG
PIC: X
DESC: A single character record flag which is set to indicate that a requisition record has or requires a MILSTRIP cancellation request to be submitted.

---

DEN:
TITLE: Cancellation Sent Date
COBOL: CANC-SENT-DATE
PIC: 9999
DESC: The Julian date of the latest MILSTRIP cancellation document submitted on a requisition record.

---

DEN:
TITLE: Change Date
COBOL: CHANGE-DATE
PIC: 9(4)
DESC: The Julian date of creation or modification of a record.

---

DEN: C003Q
TITLE: Cognizance Symbol Requisitioned
COBOL: COG-SYMBOL-REQUISITIONED
PIC: XX
DESC: Same as DEN C003 when entered on original requisition.
ORIG: DBA
CREATED: 21 JUN 72 UPDATED: 30 DEC 83
USER: UADPS-SP

---

DEN: C003E
TITLE: Condition Code

COBOL: CONDITION-CODE
PIC: A
DESC: The material condition code is a single alphabetic character which classified material in items of readiness for issue and use or to identify action underway to change the status of material. It provides the means of segmenting and identifying the physical state of material in inventory.
NOTES: Used in MISIL. COBOL PIC X; I/O - Card Code 5, P. R; AP/OP J01
Used in CAIMS as PIC X.
REFS: NAVSUP Pub 437 - Appendix 17, Logistics Management Codes.
CODES: See NAVSUP Pub 437, Appendix 17.
ORIG: DBA
CREATED: 4 JUN 76 UPDATED: 19 APR 84
USER: CAIMS LEVEL II MISIL UADPS-SP UICP

---

DEN: K020A
TITLE: Demand Code
COBOL: DEMAND-CODE
PIC: A
DESC: As specified in DEN K020.
CODES: See NAVSUP Pub 485, Supply Afloat.
ORIG: DBA
CREATED: 15 APR 71 UPDATED: 13 APR 84
USER: LEVEL II UICP

---

DEN: K023
TITLE: Distribution Code
COBOL: DISTRIBUTION-CODE
PIC: X
DESC: A code that either by itself (for other than Navy)- or in combination with the Service Designator Code- (DEN K048) (Navy only) designates the service point or activity to receive additional supply status on the requisition.
NOTES: Used in MISIL; I/O-requisitions.
REFS: Appendix 3 of NAVSUP Pub 437.
CODES: See Appendix 3 to NAVSUP Pub 437.
ORIG: DBA
CREATED: 13 JUL 76 UPDATED: 13 APR 84
USER: CAIMS IDA LEVEL II MISIL SUADPS UADPS-SP UICP

---

DEN: K001
TITLE: Document Identifier
COBOL: DOCUMENT-IDENTIFIER
PIC: XXX
DESC: 1. The document identifier code provides indentification of each document type (i.e.: requisition, passing action, status card, receipt adjustment, supply support request, etc.) in the system to which it pertains. 2. The document identifier is mandatory entry on all requisitions, supply support cataloging transactions and related documents entering the supply/cataloging distribution/information system. See SDED for other uses.
NOTES: USED in MISIL; AP/OP-C01, C10, C30, C40, C50, E10-20, J01, J10, J15, J40, J60, J8, P01. In MISIL BS record, COBOL PIC maybe XX or XXX. Used by CAIMS as PIC X(3).
REFS: NAVSUP Pub 437, Appendix 4. See SDED for others.
CODES: Refer NAVSUP Pub 437, Appendix 4.
Refer DSAR 4140.35, Change 1.

Other references per SDED.
ORIG: DBA
CREATED: 19 MAY 76 UPDATED: 23 APR 84
USER: CAIMS LEVEL II MISIL UADPS-SP UICP

---

DEN: K002B
TITLE: Document Julian Date
COBOL: DOCUMENT-JULIAN-DATE
PIC: 9999
DESC: Identifies date document or requisition was established. Consists of units digit of calender year and numeric equivalent of the day of the year (Julian Date).
NOTES: Used in CAIMS as PIC X(4).
ORIG: DBA
CREATED: 25 MAY 76 UPDATED: 24 APR 84
USER: CAIMS IDA LEVEL II MISIL SUADPS UADPS-SP UICP

---

DEN: K002
TITLE: Document Number
COBOL: DOCUMENT-NUMBER
PIC: NO PIC
DESC: A nonduplicative number constructed so as to identify the military service, the requisitioner, the Julian date of the document and the serial number. See SDED for more information.

---

DEN: K002C
TITLE: Document Number Serial Number
COBOL: DOCUMENT-NR-SERIAL-NR
PIC: XXXX
DESC: That portion of DEN K002 which applies to the serial number of the document.
NOTES: Used in CAIMS as PIC X(4).
ORIG: DBA
CREATED: 25 MAY 76 UPDATED: 24 APR 84
USER: CAIMS IDA LEVEL II MISIL SUADPS UADPS-SP UICP

---

DEN: C003C
TITLE: DOD Ammunition Code or Navy Ammunition Logistics Code
COBOL: DOD-AMMO-CODE-OR-NALC
PIC: XXXX
DESC: See SDED for complete description.
ORIG: DBA
CREATED: 1 JUL 68 UPDATED: 19 APR 84
USER: CAIMS LEVEL II UADPS-SP UICP

---

DEN:
TITLE: DOD Ammunition Code
COBOL: DODAC
PIC: NO PIC
DESC: The DODAC consists of the Federal Supply Classification and the DOD Ammunition Code or NALC. See DEN C003C for more information.

---

DEN: N603
TITLE: Due Quantity

COBOL: DUE-QUANTITY
PIC: 9(6)
DESC: The total quantity of material pre-recorded by an RSS or expediting receipt.
NOTES: Used in SUADPS; UADPS-SP; In UADPS-LEVEL II, COBOL-PIC 9(5).
ORIG: DBA
CREATED: 15 APR 71 UPDATED: 25 APR 84
USER: LEVEL II SUADPS UADPS-SP

---

DEN:
TITLE: Edit Lock
COBOL: EDIT-LOCK
PIC: X
DESC: A single character flag which is set to prevent concurrent record updating in a multi-user environment.

---

DEN: L010A
TITLE: Estimated Shipping Date
COBOL: ESTIMATED-SHIPPING-DATE
PIC: 9(4)
DESC: Date on which shipment is anticipated for status monitoring purposes. Expressed as YDDD.
NOTES: Used in TRIDENT Pre-Processor. Used in UADPS-LEVEL-II, COBOL PIC 9(5).
ORIG: 9621
CREATED: 24 AUG 79 UPDATED: 25 APR 84
USER: LEVEL II TRIDENT

---

DEN:
TITLE: Exception Flag
COBOL: EXCEPTION-FLAG
PIC: X
DESC: A single character record flag which is set to indicate that the requisition record contains exception data.

---

DEN:
TITLE: Exception Information
COBOL: EXCEPTION-INFO
PIC: X(30)
DESC: The plain language exception text data for an "AOE" requisition.

---

DEN:
TITLE: Expenditure Approving Authority
COBOL: EXPEND-APPROVING-AUTH
PIC: X(15)
DESC: The name and grade of the ordnance expenditure approving authority for an activity. This information appears in block FF of the DD Form 1348-1.

---

DEN:
TITLE: Expenditure Pending Flag
COBOL: EXPENDITURE-PENDING-FLAG
PIC: X

DESC: A single character flag which is set to indicate a lot record requiring DD Form 1348-1 expenditure document preparation.

---

DEN: C042
TITLE: Federal Supply Classification
COBOL: FEDERAL-SUPPLY-CLASSIFICATION
PIC: 9999
DESC: See DEN C001K.
ORIG: DBA
CREATED: 3 JUN 76 UPDATED: 19 APR 84
USER: LEVEL II MISIL TRIDENT UADPS-SP UICP

---

DEN:
TITLE: First Destination Address
COBOL: FIRST-DEST-ADDRESS
PIC: X(20)
DESC: The mailing address of the first destination ship-to activity. This is block 11 of the DD Form 1348-1.
NOTES: A subfield of FIRST-DESTINATION-INFO.

---

DEN:
TITLE: First Destination Information
COBOL: FIRST-DESTINATION-INFO
PIC: NO PIC
DESC: Constructed by grouping three subfields: FIRST-DEST-UIC, FIRST-DEST-NAME, and FIRST-DEST-ADDRESS. Provides the first destination shipping information.

---

DEN:
TITLE: First Destination Name
COBOL: FIRST-DESTINATION-NAME
PIC: X(15)
DESC: The activity name of the first destination address. Appears in block 11 of the DD Form 1348-1.
NOTES: A subfield of FIRST-DESTINATION-INFO.

---

DEN:
TITLE: First Destination Unit Identification Code
COBOL: FIRST-DEST-UIC
PIC: X(6)
DESC: The unit identification code of the first destination activity. Appears in block 11 of the DD Form 1348-1.
NOTES: A subfield of FIRST-DESTINATION-INFO.

---

DEN:
TITLE: Followup Flag
COBOL: FOLLOWUP-FLAG
PIC: X
DESC: A single character record flag which is set to indicate that a requisition record has or requires a MILSTRIP followup to be submitted.

---

DEN:
TITLE: Followup Sent Date
COBOL: FOLLOWUP-SENT-DATE
PIC: 9999
DESC: The Julian date of the last MILSTRIP followup document submitted for a requisition record.

---

DEN: K022
TITLE: Fund Code
COBOL: FUND-CODE
PIC: XX
DESC: The fund code is a two-column entry which may be alphabetical, numerical/alphabetical, alphabetical/numerical. It may have meaning only to the requisitioner and supplier, or may have a common meaning disseminated to all activities within a service. Fund codes are assigned for general Navy use and provide accounting information. See DEN K022 for more information.
REFS: NAVSUP Pub 437, chapter 5.
CODES: Navy assigned codes are listed in chapter 5, NAVSUP Pub 437. Standard UADPS-SP fund codes are published in NAVSUP Pub 420, chapter 8. Fund codes for other services and DOD are listed in NAVCOMPT Manual 34541.
ORIG: DBA
CREATED: 14 JUL 76 UPDATED 13 APR 84
USER: CAIMS IDA LEVEL II MISIL SUADPS UADPS-SP UICP

---

DEN:
TITLE: Hull Number
COBOL: HULL-NUMBER
PIC: X(6)
DESC: The hull number of the requisitioning activity. Consists of ship type designator (ie: DD, FF, LKA, etc.), and number.

---

DEN:
TITLE: Intermediate Unit Commander Action/Information Code
COBOL: IUC-ACT-INFO-CODE
PIC: 9
DESC: Designates type of message (ie: ATR, etc.) and distribution type (action/info) for the requisitioning activity's IUC.
CODES: Same as for ADDEE-XX-ACT-INFO-CODE.

---

DEN:
TITLE: Intermediate Unit Commander Plain Language Address
COBOL: IUC-PLAD
PIC: X(20)
DESC: The plain language communications address of the requisitioning activity's IUC.
REFS: NTP 3(F)

---

DEN:
TITLE: Immediate Superior In Command Action/Information Code
COBOL: ISIC-ACT-INFO-CODE
PIC: 9
DESC: Designates type of message (ie: ATR, etc.) and

distribution type (action/info) for the requisitioning activity's ISIC.
CODES: Same as for ADDEE-XX-ACT-INFO-CODE.

---

DEN:
TITLE: Immediate Superior In Command Plain Language Address
COBOL: ISIC-PLAD
PIC: X(20)
DESC: The plain language communications address of a requisitioning activity's ISIC.
REFS: NTP 3(F)

---

DEN:
TITLE: Last Update
COBOL: LAST-UPDATE
PIC: NO PIC.
DESC: A grouping of the USER-ID and CHANGE-DATE fields. Indicates most recent date of record modification and the user identification number of the accomplishing user.

---

DEN:
TITLE: Location
COBOL: LOCATION
PIC: X(10)
DESC: Onboard storage location of lot/serial numbered ordnance. This is the effective magazine designated by space number (ie: 3-120-0-M).

---

DEN:
TITLE: Maintenance Due Date
COBOL: MAINT-DUE-DATE
PIC: 9999
DESC: The Julian date of required maintenance for an item of ordnance.

---

DEN: K082
TITLE: Media And Status Code
COBOL: MEDIA-AND-STATUS-CODE
PIC: X
DESC: The Media and Status Code is a single character (cc-7) indicating the type of status required and the method in which it is to be furnished. See NAVSUP Pub 437, Appendix 6 for a discussion of types and media.
Used in MISIL; established by AP/OPs E10, C01.
CODES: See Appendix 6 to NAVSUP Pub 437.
ORIG: DBA
CREATED: 10 JUN 76 UPDATED: 24 APR 84
USER: CAIMS LEVEL II MISIL SUADPS UADPS-SP UICP

---

DEN: C086
TITLE: Notice Of Ammunition Reclassification Number
COBOL: NAR-NR
PIC: 9(5)
DESC: The NAR-number is comprised of the sub-elements NAR-serial (q.v.) and NAR Year (q.v.). It identi-

fies the reclassification action taken to alter the condition (hence, assets posture) of an ammunition lot-number or lot-number group.
NAR Number serves as one access key to the NXN and, hence, the MHF.

REFS: CAIMS System Elements Q219 (MHF) and Q220 (NXN)
NAVSEA Pub TW024-AA-ORD-010
OPNAVINST 5102.1 (Series)
CAIMS RS (B13.1) Elements X005, X005A, X005B

ORIG: 8512
CREATED: 9 JAN 84 UPDATED: 2 MAY 84
USER: CAIMS

---

DEN: C084
TITLE: Notice Of Ammunition Reclassification Serial
COBOL: NAR-SRL
PIC: 9(3)
DESC: One of two sub-elements comprising NAR-number. NAR-Serial serves the twofold purpose of collecting all items reclassified by a NAR action and identifying the number of reclassification actions released during a given year. Value range is as in codes, below.

REFS: CAIMS RS (B13.1) elements X005, X005A, X005B
NAVSEA Pub TWO-24-AA-ORD-010
OPNAVINST 5102.1 (Series)

CODES: Range of values is from one through 999 for a given year.

ORIG: 8512
CREATED: 9 JAN 84 UPDATED: 2 MAY 84
USER: CAIMS

---

DEN: D046D
TITLE: National Item Identification Number
COBOL: NATIONAL-ITEM-IDFCN-NR
PIC: 9(9)
DESC: A nine position non-significant number assigned by DLSC to each approved item identification under the Federal Cataloging Program.

NOTES: For PPMMS and MISIL the picture is X(9). For MISIL I/O - 5, P, R, S4, SD, 1, AP/op J01, J10, P01. The NIIN is a combination of DEN C001E (first two characters) and DEN D046 (last seven characters). The Federal Item Identification Number, DEN D046, will be replaced by DEN D046D on 30 September 1974. In UADPS - Level II, COBOL PIC X(9).
Used in CAIMS as PIC X(9).

ORIG: DBA
CREATED: 3 JUN 76 UPDATED: 19 APR 84
USER: CAIMS LEVEL II MISIL TRIDENT UADPS-SP UICP

---

DEN:
TITLE: Nick Name
COBOL: NICK-NAME
PIC: X(10)
DESC: A user-defined short title description for an item of ordnance.

---

DEN:
TITLE: Nomenclature
COBOL: NOMENCLATURE
PIC: X(30)

DESC: The complete or abbreviated long title for an item of ordnance (ex.: MK 46 MOD 5 NEARTIP TORPEDO).

---

DEN:
TITLE: Order Quantity
COBOL: ORDER-QUANTITY
PIC: 9(5)
DESC: The total original quantity of an ordnance item placed under a single requisition.

---

DEN:
TITLE: Partial Order Information
COBOL: PARTIAL-ORDER-INFO
PIC: NO PIC
DESC: Contains four subfields: RECEIPT-DATE, RECEIVED-QUANTITY, BALANCE-QUANTITY and BALANCE-ESD. Describes the remaining balance quantity of a partially filled requisition.

---

DEN:
TITLE: Phone
COBOL: PHONE
PIC: X(4)
DESC: Provides the telephone number for a user.

---

DEN:
TITLE: Password
COBOL: PASSWORD
PIC: X(6)
DESC: A user-defined code word used in conjunction with USER-ID to gain system access.
NOTES: A subfield of ACCESS-VECTOR.

---

DEN: E606C
TITLE: Priority Code, Other Cog
COBOL: PRIORITY-CODE-OTHER-COG
PIC: 99
DESC: Same as DEN K025, except that three different codes may be input for cog differences. This element applies to all COG items except "IR" COG and APA funded items.
REFS: E60
ORIG: 9661
CREATED: 1 JUN 81 UPDATED: 24 JUN 82
USER: UICP

---

DEN: K024
TITLE: Project Code
COBOL: PROJECT-CODE
PIC: XXX
DESC: A specific code assigned by a military service or the Department of Defense to identify a specific project of a general or special program nature. See DEN K024 for more information.
NOTES: Established by AP/OPs E10 and C01. Updated by AP/OP C10.
Used in CAIMS as PIC X(3).
REFS: Appendix 8 of NAVSUP Pub 437.

CODES: See Appendix 8 of NAVSUP Pub 437 for codes.
ORIG: DBA
CREATED: 10 JUN 76 UPDATED: 13 APR 84
USER: CAIMS LEVEL II MISIL SUADPS UADPS-SP UICP

---

DEN:
TITLE: Rank
COBOL: RANK
PIC: X(5)
DESC: A four character abbreviation for a user's rank or rating (ie.: LCDR, BM1, STGCS, etc.).

---

DEN: K318
TITLE: Receipt Date
COBOL: RECEIPT-DATE
PIC: 9999
DESC: The Julian date on which material is received.
NOTES: Date on which railroad car was spotted or highway shipper arrives aboard station. (Also called Tailgate Date.)
ORIG: DBA
CREATED: 10 MAR 72 UPDATED: 13 APR 84
USER: CAIMS IDA

---

DEN:
TITLE: Received Quantity
COBOL: RECEIVED-QUANTITY
PIC: 9(7)
DESC: The quantity reported received by the user under a given requisition.

---

DEN:
TITLE: Reorder Flag
COBOL: REORDER-FLAG
PIC: X
DESC: An ammunition record field which is set to indicate that a reorder is required for a given NALC.

---

DEN:
TITLE: Reorder Quantity
COBOL: REORDER-QTY
PIC: 9(5)
DESC: Indicates the quantity of a given NALC to be reordered. It is either provided by the user directly (ie.: item reorder) or is calculated by: ALLOW-QTY - TOTAL-DODAC-QTY-ONHAND + DUE-QUANTITY (ie.: global reorder).

---

DEN: C877A
TITLE: Required Delivery Date
COBOL: REQUIRED-DELIVERY-DATE
PIC: xxx
DESC: Represents the date that the material is required is required by the submitting country/activity. The RDD is received from DSAA on the Card Code 5 when ordering material and services. MISIL programs convert the RDD to the MILSTRIP system when requisitions are prepared. The first digit of the

RDD contains the last position of the calender year The second and third digits of the RDD contain the month of the calender year.
NOTES: Used in MISIL; I/O - Card Code 5, P, R; AP/OP-J01.
REFS: MASM, Part II, APP A-20.
ORIG: DBA
CREATED: 12 MAY 76 UPDATED: 12 MAY 76
USER: MISIL

---

DEN:
TITLE: Responsible Work Center
COBOL: RESPONSIBLE-WORK-CENTER
PIC: X(4)
DESC: The cognizant work center code for a given ordnance item (NALC).

---

DEN:
TITLE: Requisition Outstanding Flag
COBOL: REQN-OUTSTANDING-FLAG
PIC: X
DESC: A single character record flag which is set to indicate that a requisition record has or requires a MILSTRIP requisition to be submitted.

---

DEN: K002A
TITLE: Requisitioner Identification Code
COBOL: RQNR-IDENTIFICATION-CODE
PIC: X(5)
DESC: Accounting number or code which identifies the activity, operational unit, or agency submitting or originating a document, or the whom the document is established.
NOTES: In MISIL Transactions: COBOL Picture Changed - X(6)
REFS: NAVCOMPT Manual Vol. II, Chapter 5 DODAAD (Activity Address Directory of the Department of Defense).
ORIG: DBA
CREATED: 6 NOV 73 UPDATED: 24 APR 84
USER: LEVEL II MISIL UICP

---

DEN: A001C
TITLE: Routing Identifier-Last Holding Activity
COBOL: RTNG-IDR-LAST-HOLDING-ACTVY
PIC: AXX
DESC: DEN A001 when used specifically to identify the activity to which the MILSTRIP Document wa passed or referred.
ORIG: DBA
CREATED: 21 JUN 72 UPDATED: 24 APR 84
USER: LEVEL II UADPS-SP UICP

---

DEN: C029
TITLE: Shelf Life Action Code
COBOL: SHELF-LIFE-ACTION-CODE
PIC: XX
DESC: A code denoting the action to be taken for an item at the expiration of the shelf life period indicated by the Shelf Life Code, DEN C028.
CODES: See Appendix 17 to NAVSUP Pub 437.
ORIG: DBA
CREATED: 1 SEP 68 UPDATED: 19 APR 84

USER: CAIMS LEVEL II SUADPS TRIDENT UADPS-SP UICP

---

DEN: C028
TITLE: Shelf Life Code
COBOL: SHELF-LIFE-CODE
PIC: X
DESC: A code denoting the shelf life span of material from the date of manufacture of previous inspection to the date of test for continued usefulness or disposition.
CODES: See Appendix 17 to NAVSUP Pub 437.
ORIG: DBA
CREATED: 12 NOV 75 UPDATED: 19 APR 84
USER: CAIMS LEVEL II SUADPS TRIDENT UADPS-SP UICP

---

DEN:
TITLE: Shelf Life Information
COBOL: SHELF-LIFE-INFO
PIC: NO PIC
DESC: Contains two subfields: SHELF-LIFE-CODE and SHELF-LIFE-ACTION-CODE. Describes storage monitoring requirements for a given ordnance item.

---

DEN:
TITLE: Ship To Information
COBOL: SHIP-TO-INFO
PIC: NO PIC
DESC: Contains two subfields: SHIP-TO-UIC and SHIP-TO-NAME. Identifies the ship to addressee in block "B" of the DD Form 1348-1 expenditure document.

---

DEN:
TITLE: Ship To Name
COBOL: SHIP-TO-NAME
PIC: X(15)
DESC: The plain language activity name of the ship to addressee.
NOTES: A subfield of SHIP-TO-INFO.

---

DEN:
TITLE: Ship To Unit Identification Code
COBOL: SHIP-TO-UIC
PIC: X(6)
DESC: The unit identification code of the ship to addressee.
NOTES: A subfield of SHIP-TO-INFO.
REFS: NAVCOMPT Manual Vol. 2, Chapter 5.

---

DEN: K021
TITLE: Signal Code
COBOL: SIGNAL-CODE
PIC: X
DESC: The purpose of the signal code is twofold. This code designates the fields (card columns) which contain the intended consignee (ship to) and the activity to receive the bills and effect payment (bill to), when applicable. The "bill to" activity for intra-Navy transactions also may indicate the

chargeable or accountable activity. All requisitions and documents resulting therefrom will contain the appropriate signal code.
NOTES: UADPS-SP. On the Billing Cross Reference and NSF/RIS Allotment Ledger Files, this code represents the signal code cited on an original request document. Used in CAIMS as PIC X in DTN, FTD, DCT, RSF files.
CODES: See Appendix 12 to NAVSUP Pub 437.
ORIG: DBA
CREATED: 6 NOV 73 UPDATED: 13 APR 84
USER: CAIMS IDA LEVEL II MISIL SUADPS UADPS-SP UICP

---

DEN:
TITLE: Status Code
COBOL: STATUS-CODE
PIC: XX
DESC: The latest status recorded on a requisition.
CODES: See NAVSUP Pub 485 for codes.

---

DEN:
TITLE: Status Date
COBOL: STATUS-DATE
PIC: 999
DESC: The date on which the latest status was recorded for a given requisition.

---

DEN:
TITLE: Status Information
COBOL: STATUS-INFO
PIC: NO PIC
DESC: Contains four subfields: STATUS-CODE, RTNG-IDR-LAST-HOLDING-ACTVY, ESTIMATED-SHIPPING-DATE and STATUS-DATE. Provides the most recent information concerning the processing status of a requisition.

---

DEN:
TITLE: Total DOD Ammunition Code Quantity Onhand
COBOL: TOTAL-DODAC-QTY-ONHAND
PIC: 9(5)
DESC: Indicates the total quantity of an ordnance item (NALC) onhand. It is the sum of individual lots/serialized items for a given NALC.

---

DEN:
TITLE: Total Lot Quantity Onhand
COBOL: TOTAL-LOT-QTY-ONHAND
PIC: 9(5)
DESC: Indicates the total lot quantity for a given ordnance item. For a serialized assembled ordnance item, this is equal to one (1).

---

DEN:
TITLE: Transaction Information
COBOL: TRANSACTION-INFO
PIC: NO PIC
DESC: Contains five subfields: ATR-TRNSN-QTY, ATR-TRNSN-CODE, ATR_TRNSN-FLAG, ATR-TRNSN-DATE and ATR-

CONDITION-CODE. Provides information pertaining to a transaction involving a lot or serialized item and supports ATR preparation.

---

DEN:
TITLE: Transaction Type Code
COBOL: TRANSACTION-TYPE-CODE
PIC: X
DESC: A single character code which identifies a transaction record as either a receipt, expenditure, or loss or gain by inventory.
CODES: R: Receipt
E: Expenditure
L: Loss by inventory
G: Gain by inventory

---

DEN:
TITLE: Type Commander Action/Information Code
COBOL: TYCOM-ACT-INFO-CODE
PIC: 9
DESC: Designates type of message (ATR, etc.) and distribution type (action/information) for the requisitioning activity's type commander.
CODES: Same as for ADDEE-XX-ACT-INFO-CODE.

---

DEN:
TITLE: Type Commander Plain Language Address
COBOL: TYCOM-PLAD
PIC: X(25)
DESC: The plain language communications address for a requisitioning activity's type commander.
REFS: NTP 3(F)

---

DEN:
TITLE: Unit Information
COBOL: UNIT-INFO
PIC: NO PIC
DESC: Contains four subfields: UNIT-NAME, HULL-NUMBER, RONR-IDENTIFICATION-CODE and UNIT-PLAD. Provides the requisitioning activity's name, hull number, unit identification code and plain language address

---

DEN:
TITLE: Unit Name
COBOL: UNIT-NAME
PIC: X(20)
DESC: The requisitioning activity's name (ex.: USS SAMPSON).

---

DEN: C005
TITLE: Unit Of Issue
COBOL: UNIT-OF-ISSUE
PIC: AA
DESC: An abbreviation which represents a determinate amount or quantity and serves as a unit of measurement when issuing the item.
NOTES: COBOL PIC in MISIL is XX. In MISIL, Grant Aid Program Dollar Lines received on DSAA Card Code "5" inputs are identified by the presence of "XX" in

the Unit of Issue field. Codes preceded by an asterisk (*) are not definitive in accordance with the Federal Manual for Supply Cataloging M1-7. Used in CAIMS as PIC X(8) in MCI and PIC XX in DCT, RSF, DTN, PPS, MAF, MDF Files.
CODES: See Appendix 23 to NAVSUP Pub 437.
ORIG: DBA
CREATED: 3 JUN 76 UPDATED: 19 APR 84
USER: CAIMS LEVEL II MISIL SUADPS TRIDENT UADPS-SP UICP

---

DEN:
TITLE: Unit Plain Language Address
COBOL: UNIT-PLAD
PIC: X(15)
DESC: The requisitioning activity's message communication plain language address.
REFS: NTP 3(F)

---

DEN: B053
TITLE: Unit Price
COBOL: UNIT-PRICE
PIC: 9(7)V99
DESC: The price of the individual item of supply per unit of issue.
NOTES: UADPS-LEVEL II (Appl. M): COBOL PIC 9(5)V99. MISIL PIC is S9(10)V99. TRIDENT-A/O T24.
TRIDENT S-TE: COBOL PIC is 9(5).
Used in CAIMS as follows:
IOF 9(8)V99
9(5)V99
PPS 9(5)V99
DCT-MAF-MDFX(7)
FV EXTRACT 9(7)V9(8)
COBOL name used in UADPS-SP Demand History File is UNIT-PRICE-7, with PIC 9(5)V99.
ORIG: DBA
CREATED: 21 MAY 76 UPDATED: 16 APR 84
USER: IDA

---

DEN:
TITLE: User Identification Code
COBOL: USER-ID
PIC: XXX
DESC: A three character unique code that identifies a system user.
NOTES: A subfield of LAST-UPDATE and ACCESS-VECTOR.

---

DEN:
TITLE: User Name
COBOL: USER-NAME
PIC: X(15)
DESC: The actual system user's name.

---

DEN: E902A
TITLE: Work Center
COBOL: WORK-CENTER
PIC: X(4)
DESC: A code used to identify an organizational subdivision. The code may be used to identify repair work centers having primary responsibility for key

operation completion or work centers assisting in the accomplishment of key operations.

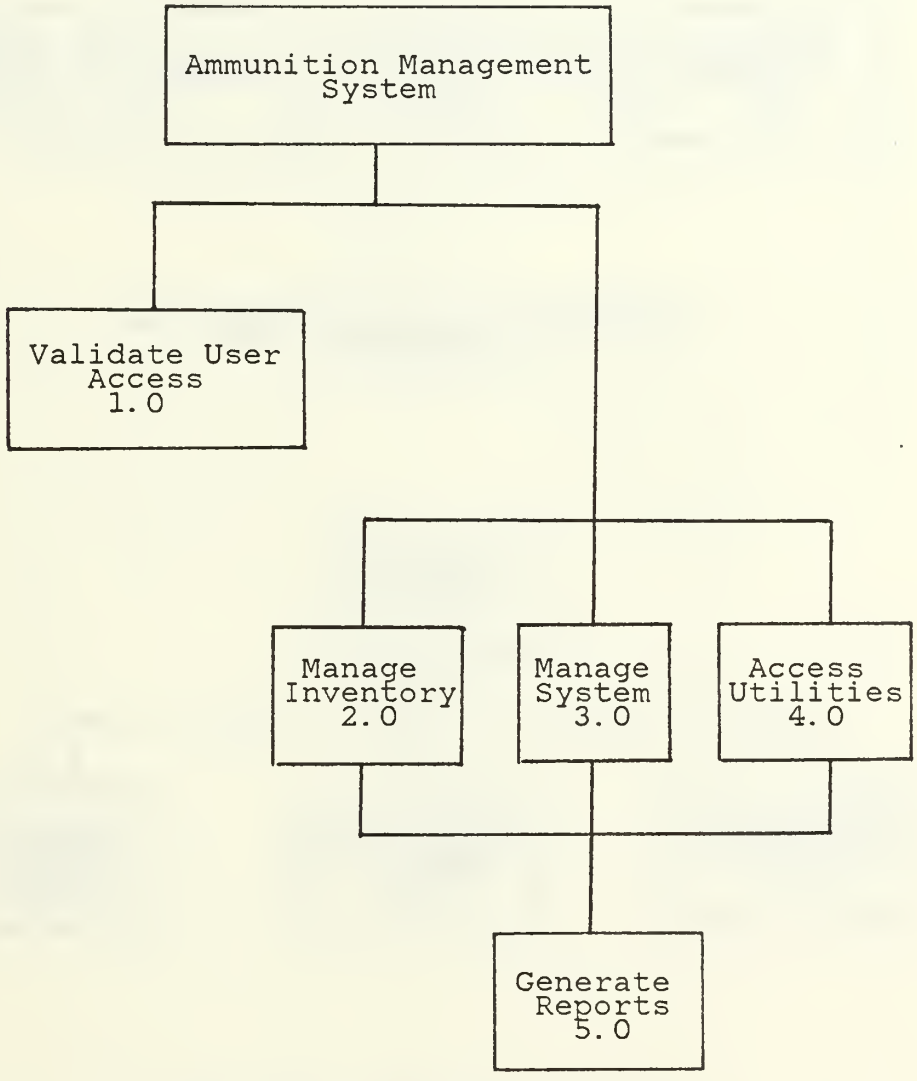
NOTES: (1) TRIDENT Logistic Data System. (2) A/O T22 TRIDENT - A/O M24 - Represents Calibration Work Center Also, COBOL Name is "CLBRN-WORK-CENTER".

ORIG: 9621

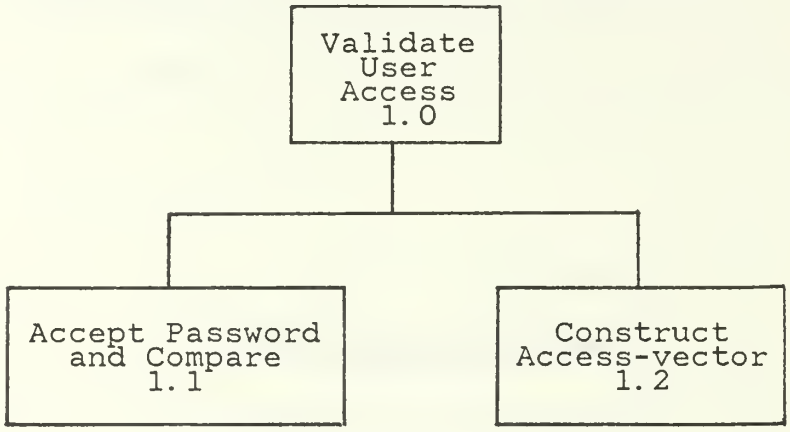
CREATED: 15 NOV 79 UPDATED: 20 FEB 80

USER: TRIDENT UICP

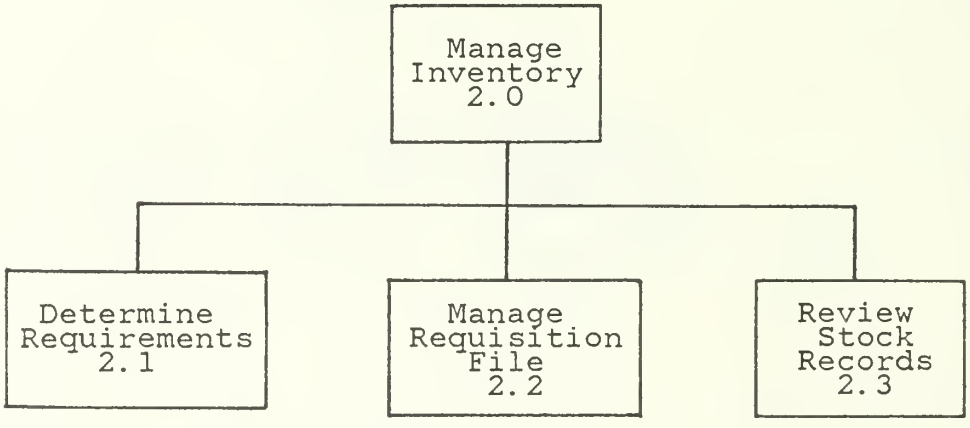
APPENDIX C
Program Structure Diagrams
(Through Second Level Refinement)



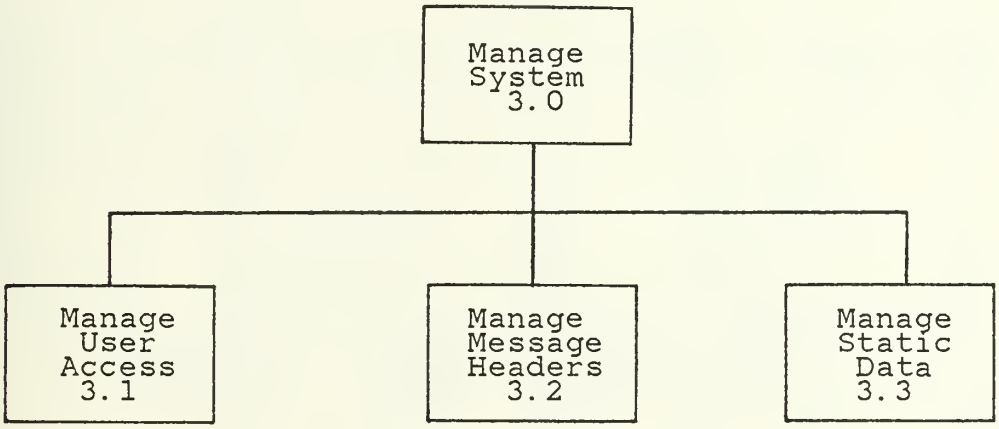

Major Subsystem (Program) Structure



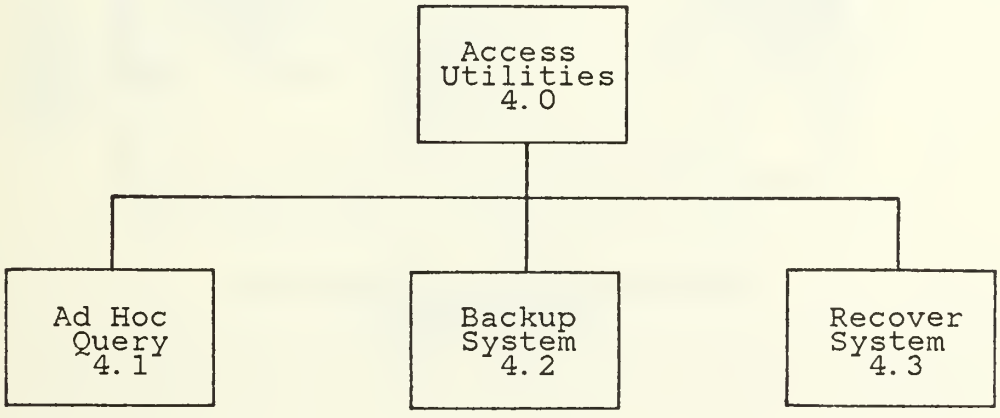

Validate User Access Subsystem Structure



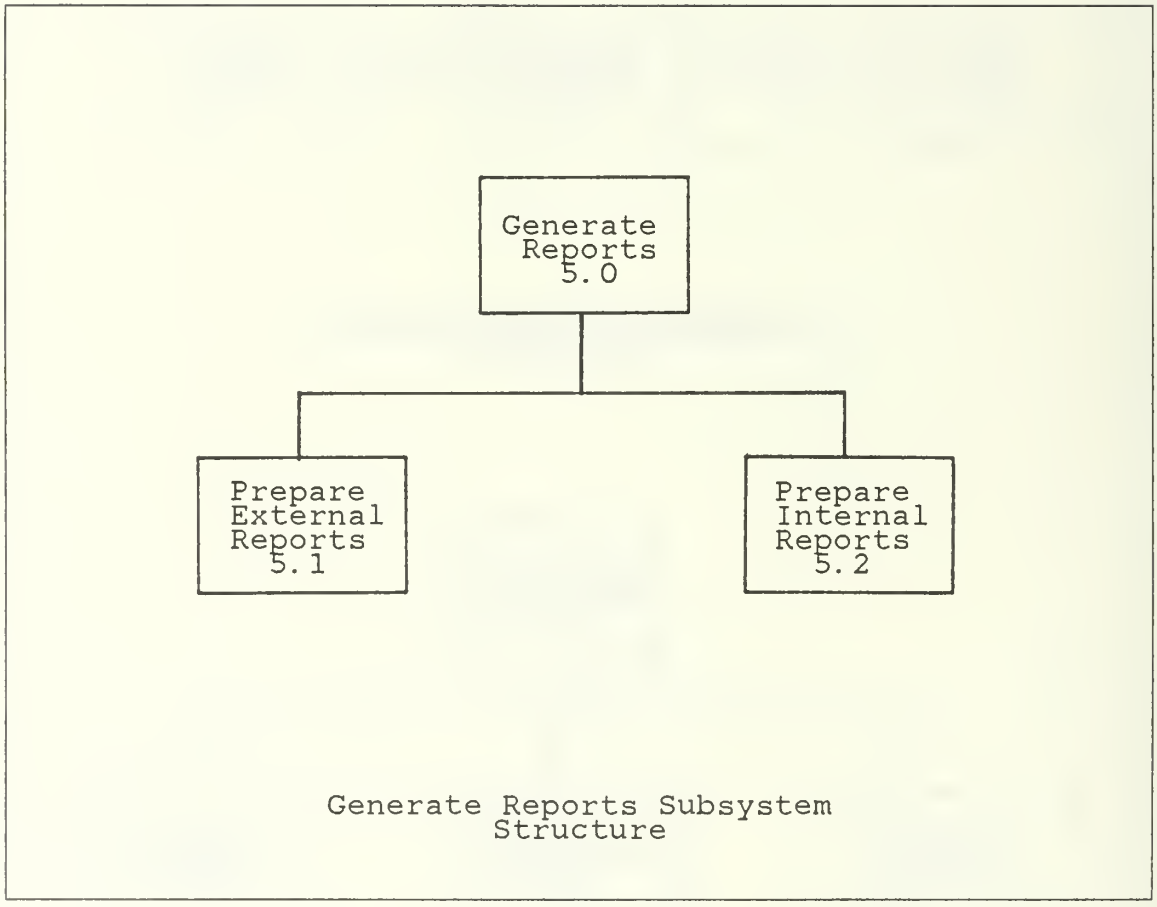

Manage Inventory Subsystem Structure




Manage System Subsystem Structure

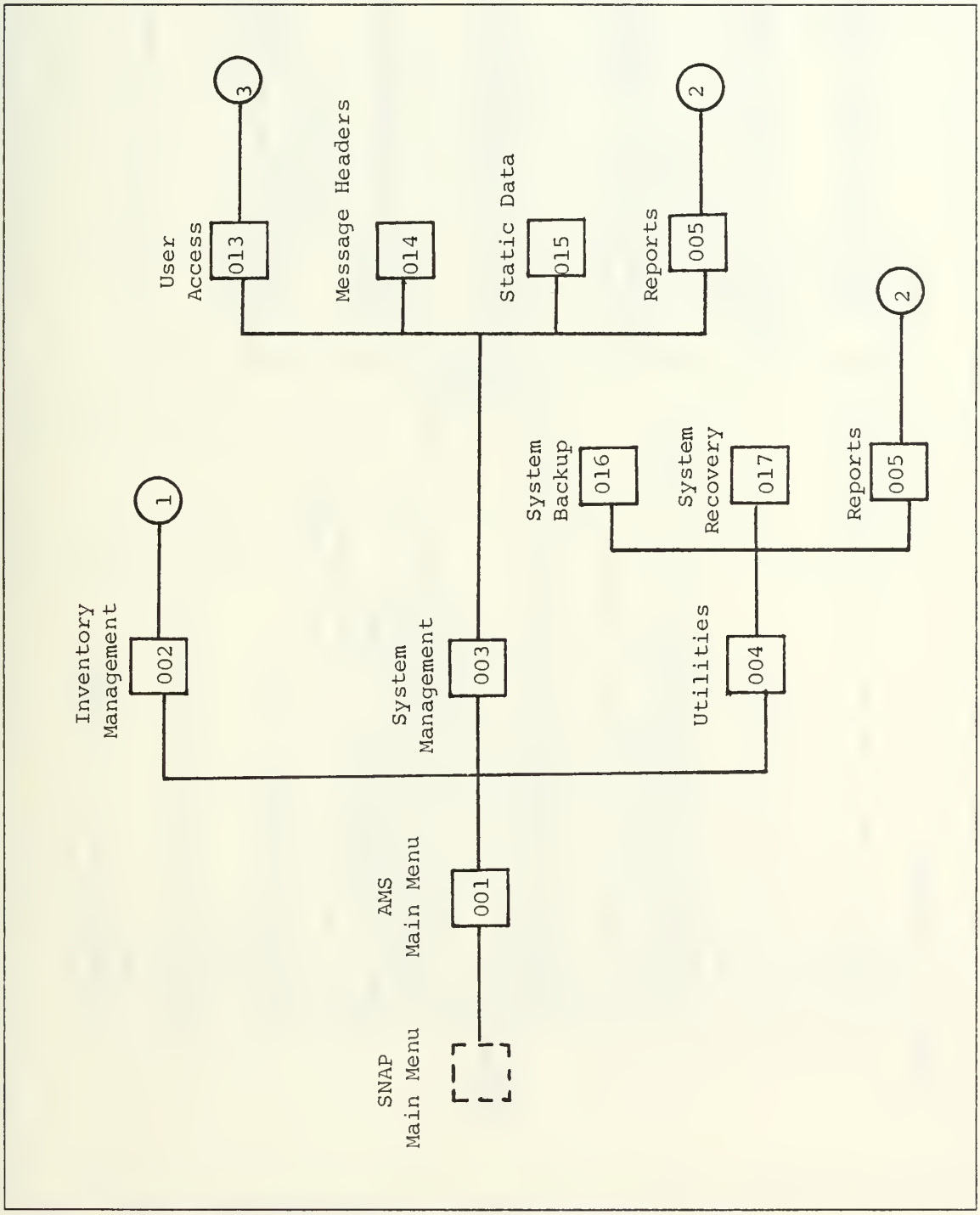


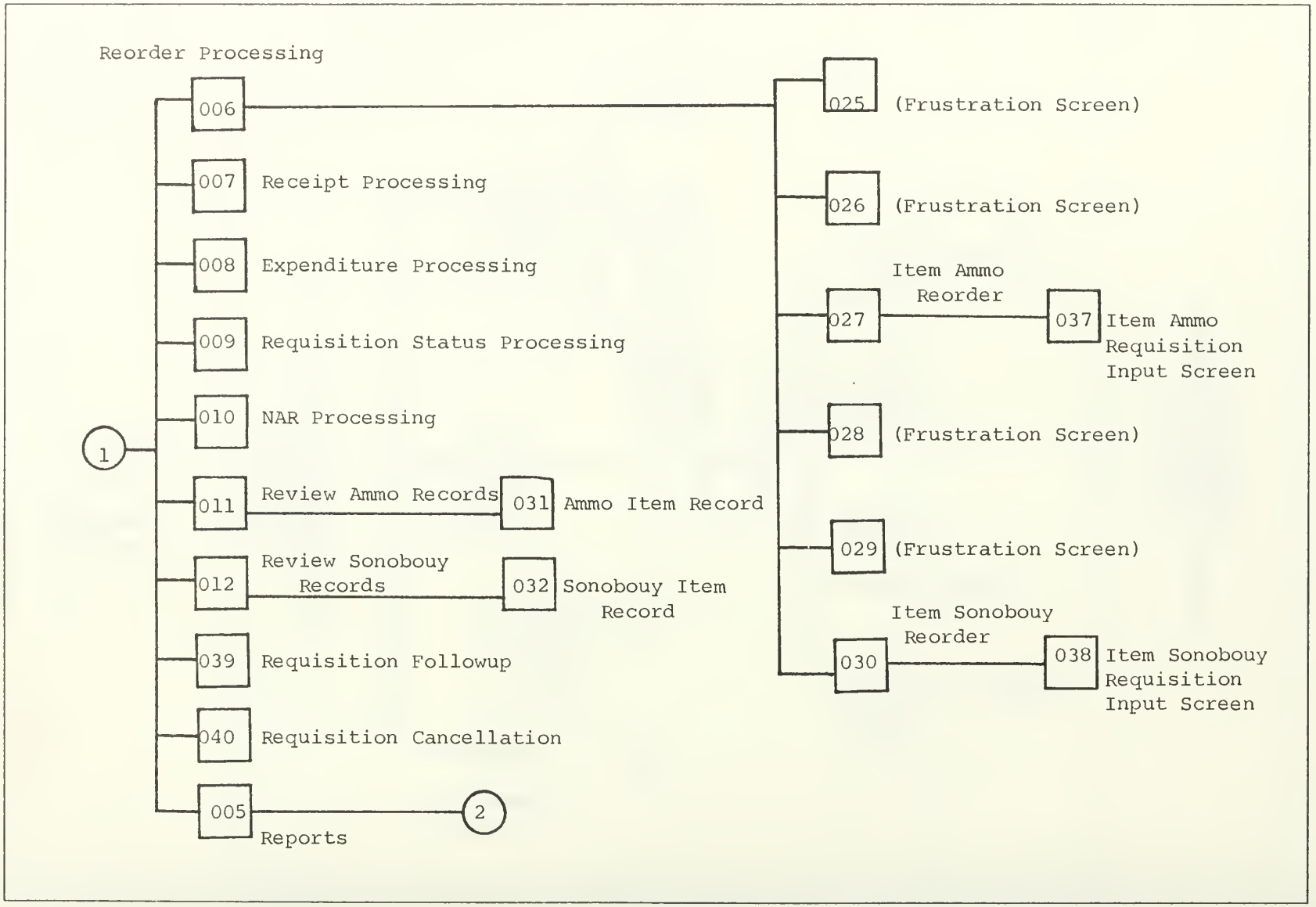

Access Utilities Subsystem Structure

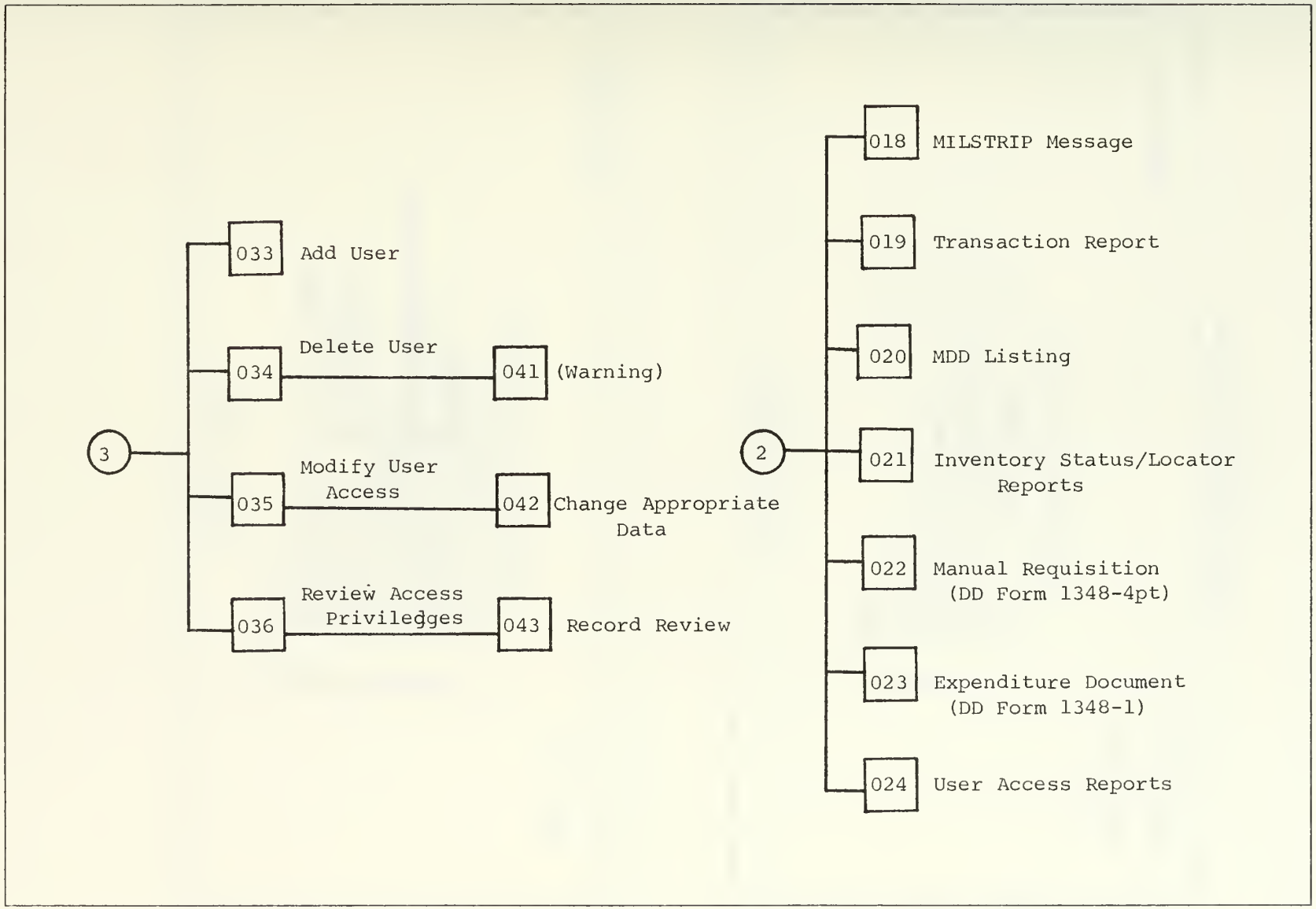



Generate Reports Subsystem Structure

APPENDIX D
Menu Screen Structure
Diagram




Reorder Processing
006
007 Receipt Processing
008 Expenditure Processing
009 Requisition Status Processing
010 NAR Processing
1
011 Review Ammo Records
031 Ammo Item Record
012 Review Sonobouy Records
032 Sonobouy Item Record
039 Requisition Followup
040 Requisition Cancellation
005 Reports
2
025 (Frustration Screen)
026 (Frustration Screen)
Item Ammo Reorder
027
037 Item Ammo Requisition Input Screen
028 (Frustration Screen)
029 (Frustration Screen)
Item Sonobouy Reorder
030
038 Item Sonobouy Requisition Input Screen


018 MILSTRIP Message
019 Transaction Report
020 MDD Listing
2
021 Inventory Status/Locator Reports
022 Manual Requisition (DD Form 1348-4pt)
023 Expenditure Document (DD Form 1348-1)
024 User Access Reports
033 Add User
Delete User
034
041 (Warning)
3
Modify User Access
035
042 Change Appropriate Data
Review Access Privileges
036
043 Record Review

APPENDIX E
Menu Screen Formats

AMS 001

MAIN MENU

1. Inventory Management
2. System Management
3. Utilities
4. Exit

PF 13: Help

AMS 002

INVENTORY MANAGEMENT

1. Reorder Processing
2. Receipt Processing
3. Expenditure Processing
4. Requisition Status Processing
5. NAR Processing
6. Requisition Followup
7. Requisition Cancellation
8. Review Ammo Records
9. Review Sonobouy Records
10. Reports
11. Exit

PF 13: Help

AMS 003

SYSTEM MANAGEMENT

1. User Access
2. Message Headers
3. Static Data
4. Reports
5. Exit

PF 13: Help

AMS 004

UTILITIES

1. Ad Hoc Query
2. System Backup
3. System Recovery
4. Reports
5. Exit

PF 13: Help

AMS 005

REPORTS

1. MILSTRIP Message
2. Transaction Report
3. MDD Listing
4. Inventory Status/Locater Reports
5. Manual Reqn (DD 1348 4 part)
6. Expenditure Document (DD 1348-1)
7. User Access Reports
8. Exit

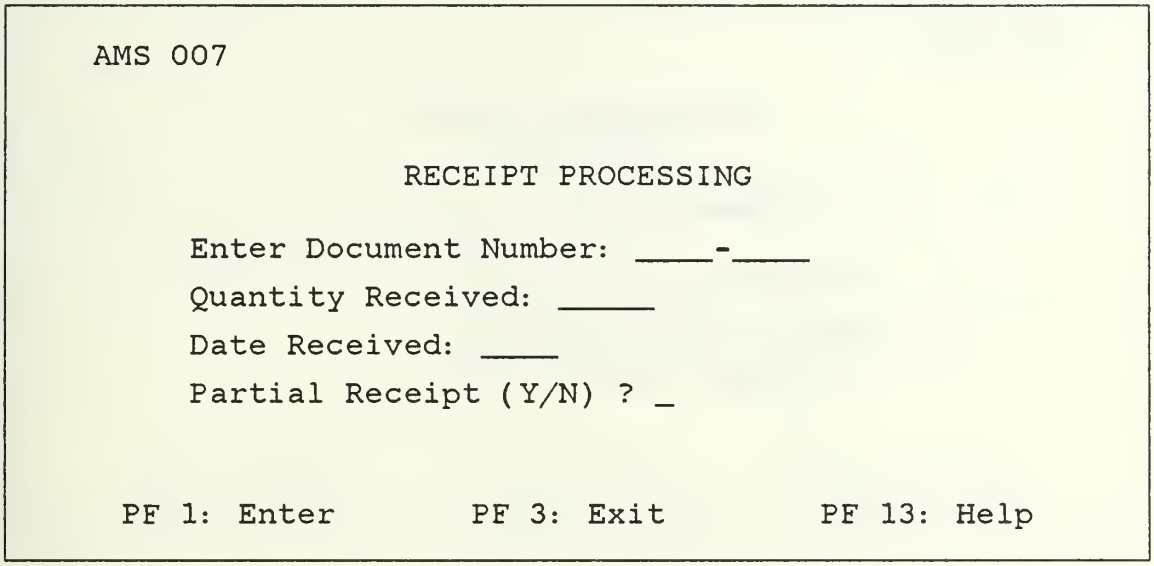
PF 13: Help

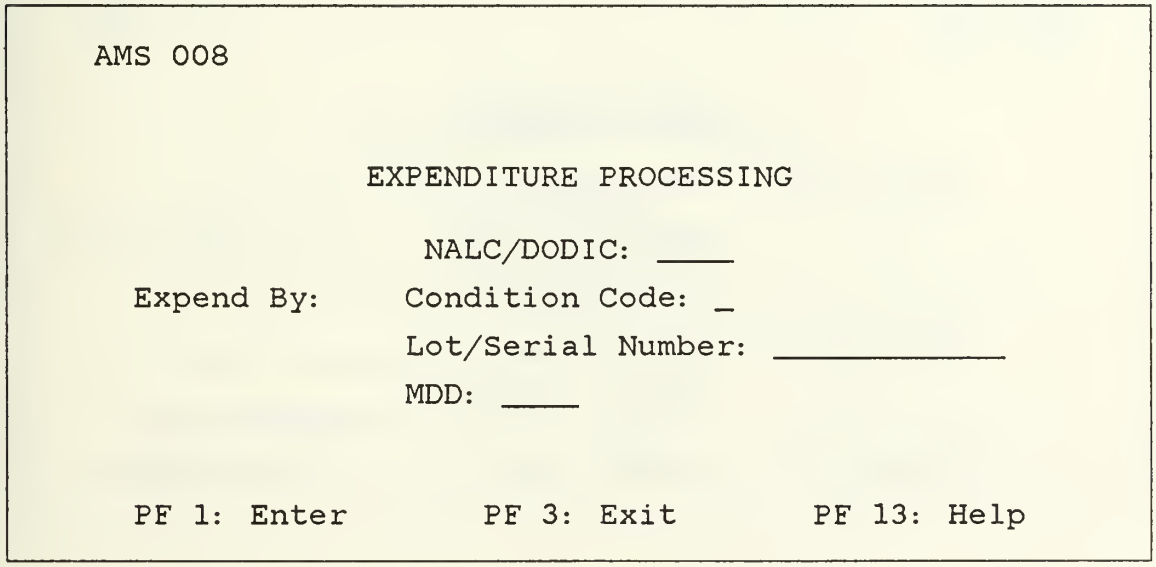
AMS 006

REORDER PROCESSING

1. Global Ammo Reorder
2. Trial Global Ammo Reorder
3. Item Ammo Reorder
4. Global Sonobouy Reorder
5. Trial Sonobouy Reorder
6. Item Sonobouy Reorder
7. Exit

PF 13: Help


AMS 007
RECEIPT PROCESSING
Enter Document Number: ____-____
Quantity Received: _____
Date Received: ____
Partial Receipt (Y/N) ? _
PF 1: Enter
PF 3: Exit
PF 13: Help


AMS 008
EXPENDITURE PROCESSING
NALC/DODIC: ____
Expend By:
Condition Code: _
Lot/Serial Number: _____
MDD: ____
PF 1: Enter
PF 3: Exit
PF 13: Help

AMS 009

REQUISITION STATUS
PROCESSING

Document Number: _____-_____

Status: __

Requisition Holder: ___

Estimated Shipping Date: ___

PF 1: Enter PF 3: Exit PF 13: Help

AMS 010

NAR PROCESSING

Enter One: NALC/DODIC: _____

Nomenclature: _____

Nick Name: _____

Enter One: Lot Number: _____

Serial Number: _____

PF 1: Enter PF 3: Exit PF 13: Help

AMS 011

REVIEW AMMO RECORDS

Enter One: NALC/DODIC: _____
Nomenclature: _____
Nick Name: _____

PF 1: Enter PF 3: Exit PF 13: Help

AMS 012

REVIEW SONOBOUY RECORDS

Enter One: NALC/DODIC: _____
Nomenclature: _____
Nick Name: _____

PF 1: Enter PF 3: Exit PF 13: Help

AMS 013

USER ACCESS

1. Add User
2. Delete User
3. Modify User Access
4. Review Access Privileges
5. Exit

PF 13: Help

AMS 014

MESSAGE HEADERS

Unit PLAD: _____

| | | Act/Info Code |
|---|---|---|
| TYCOM PLAD: | _____ | _ |
| IUC PLAD: | _____ | _ |
| ISIC PLAD: | _____ | _ |
| Addee 01 PLAD: | _____ | _ |
| 02 PLAD: | _____ | _ |
| 03 PLAD: | _____ | _ |
| 04 PLAD: | _____ | _ |
| 05 PLAD: | _____ | _ |
| 06 PLAD: | _____ | _ |

PF 1: Enter PF 3: Exit PF 13: Help

AMS 015

STATIC DATA

Unit Name: _____ Hull Number: _____

UIC: _____ Unit PLAD: _____

Expenditure Approving Authority: _____

Ship To:
UIC: _____ Name: _____

First Destination:
UIC: _____ Name: _____

PF 1: Enter PF 3: Exit PF 13: Help

AMS 016

SYSTEM BACKUP

1. Insert a formatted diskette into Drive B.
2. Press PF 5.

* Note *
Keyboard is locked during backup operation

3. Remove diskette from Drive B. Label and store in a safe place.
4. System will return to the Utility Menu when backup is completed.

PF 2: Quit PF 5: Backup PF 13: Help

AMS 017

SYSTEM RECOVERY

1. Insert Backup Diskette into Drive A.
2. Press PF 6.

* Note *
Keyboard is locked during recovery operation

3. System returns to the Utilities Menu when recovery is completed.

PF 2: Quit PF 6: Recover PF 13: Help

AMS 018

MILSTRIP MESSAGE

1. Ammo DAAS MILSTRIP Requisition
2. Ammo Exception MILSTRIP Requisition
3. Sonobouy DAAS MILSTRIP Requisition
4. Sonobouy Exception MILSTRIP Requisition
5. Requisition Followup
6. Requisition Modifier
7. Requisition Cancellation
8. Exit

PF 13: Help

AMS 019

TRANSACTION REPORT

1. Ammunition
2. Sonobouy
3. Exit

PF 13: Help

AMS 020

MDD LISTING

Enter One:

NALC: ____ Lot/Serial: _____

Or:

MDD Cut-Off Date: ____

PF 2: Quit PF 13: Help

AMS 021

INVENTORY STATUS/LOCATOR
REPORTS

1. Outstanding Requisition Listing
2. Pending Requisition Listing
3. Pending Expenditure Listing
4. Ammo Master Stock Record Card
5. Ammo Lot/Location Card
6. Ammo Serial/Location Card
7. Sonobouy Location Card
8. Exit

PF 13: Help

AMS 022

MANUAL REQN (DD 1348 4-PART)

Select One (x): _ Global
_ NALC/DODIC: _____
_ Document NR: _____
_ From NR: _____ to _____

PF 1: Enter PF 2: Quit PF 3: Exit PF 13: Help

```
AMS 023

                EXPENDITURE DOCUMENT (1348-1)

        Select One (x): _ Global
                        _ NALC/DODIC: ____
                        _ Lot/Serial NR: _________

PF 1: Enter    PF 2: Quit    PF 3: Exit    PF 13: Help
```

```
AMS 024

                    USER ACCESS REPORTS

        Select One (x): _ Global
                        _ Name: ____________________
                        _ User ID: ___

           Display Password in Report (Y/N) ? _

PF 1: Enter    PF 2: Quit    PF 3: Exit    PF 13: Help
```

AMS 025

GLOBAL AMMUNITION REORDER
IN PROGRESS

* Note *
Keyboard is locked

System will return to the Reorder
Processing Menu when complete

AMS 026

TRIAL GLOBAL AMMUNITION REORDER
IN PROGRESS

* Note *
Keyboard is locked

System will return to the Reorder
Processing Menu when complete.

AMS 027

ITEM AMMO REORDER

NALC: _____

PF 1: Enter  PF 3: Exit  PF 13: Help

AMS 028

GLOBAL SONOBOUY REORDER
IN PROGRESS

* Note *
Keyboard is locked

System will return to the Reorder
Processing Menu when complete.

AMS 029

TRIAL GLOBAL SONOBOUY REORDER
IN PROGRESS

* Note *
Keyboard is locked

System will return to the Reorder
Processing Menu when complete.

AMS 030

ITEM SONOBOUY REORDER

NALC: _____

PF 1: Enter  PF 2: Quit  PF 3: Exit  PF 13: Help

AMS 031

AMMUNITION ITEM RECORD

Nomenclature: * Nick Name: *
Allowance List: * NALC/DODIC: *
NSN: * COG: *
Lot/Serial: * MDD: * Location: *
Responsible Work Center: * Unit Price: *

Quantities:
Allowed: * On Hand: * On Order: *

Outstanding Requisitions:

* * *

Last Updated On: * By: *

PF 3: Exit PF 7: Last Record PF 8: Next Record
PF 13: Help

AMS 032

SONOBOUY ITEM RECORD

Nomenclature: * Nick Name: *
Allowance List: * NALC/DODIC: *
NSN: * COG: *
Lot/Serial: * MDD: * Location: *
Responsible Work Center: * Unit Price: *

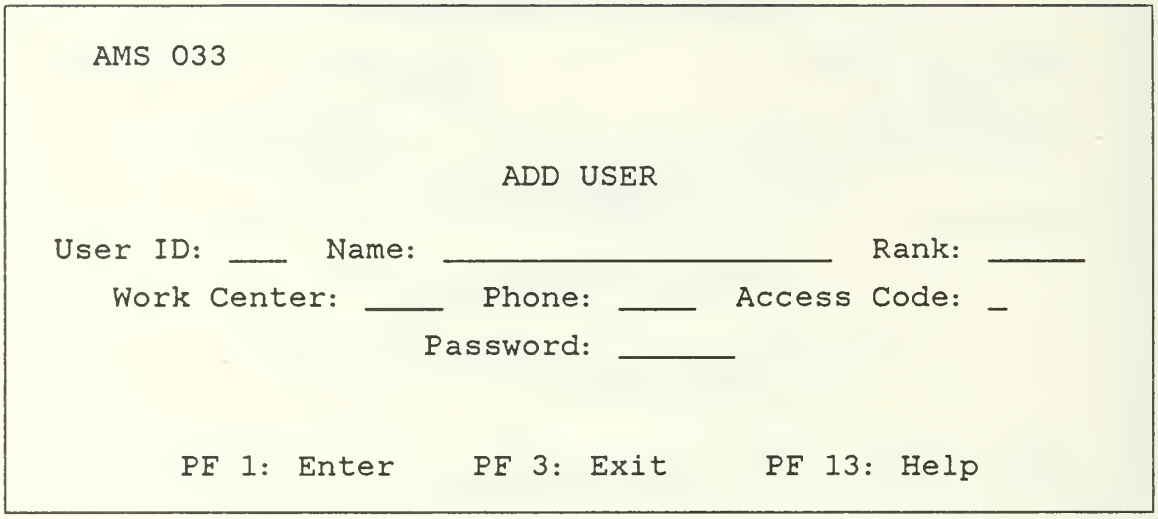
Quantities:
Allowed: * On Hand: * On Order: *

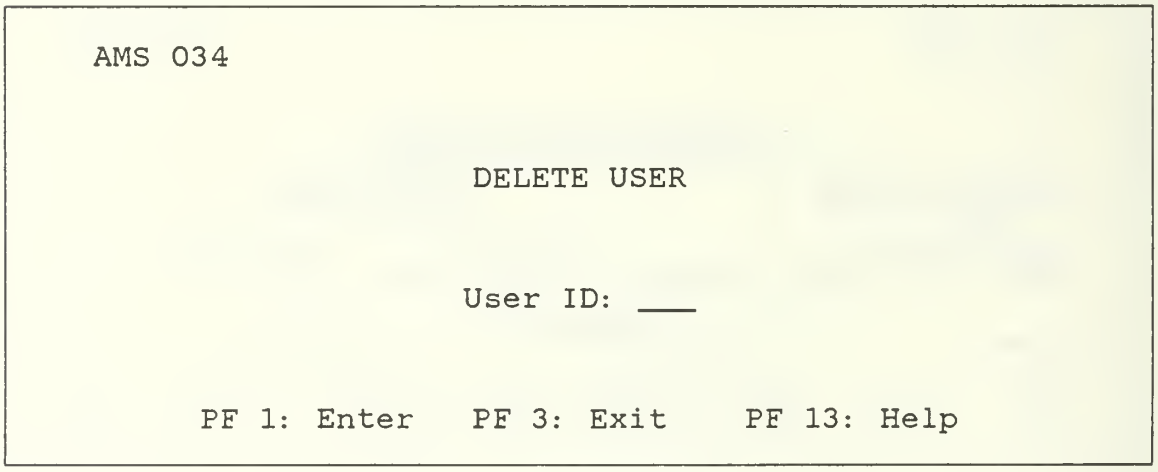
Outstanding Requisitions:
* * *

Last Updated On: * By: *

PF 3: Exit PF 7: Last Record PF 8: Next Record
PF 13: Help

* Data Field Pre-Filled By Software


AMS 033
ADD USER
User ID: ___ Name: _____ Rank: _____
Work Center: _____ Phone: _____ Access Code: _
Password: _____
PF 1: Enter PF 3: Exit PF 13: Help


AMS 034
DELETE USER
User ID: ___
PF 1: Enter PF 3: Exit PF 13: Help

AMS 035

MODIFY USER ACCESS

User ID: ____

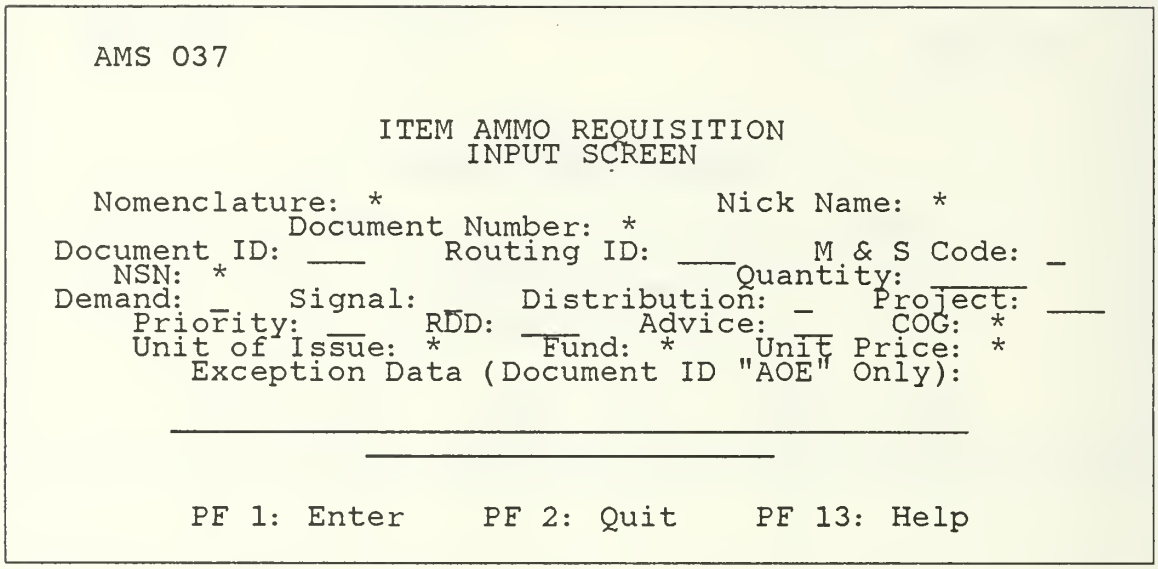
PF 1: Enter  PF 3: Exit  PF 13: Help

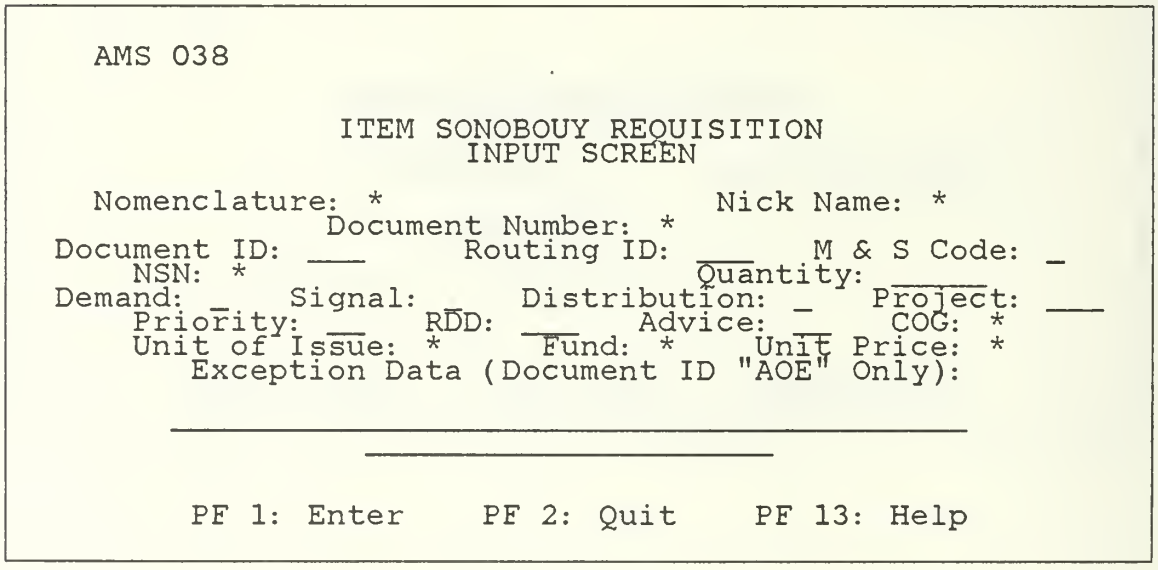
AMS 036

REVIEW ACCESS PRIVILEGES

Select One (x):  _ First Record
_ User ID: ____
_ Name: _____

PF 1: Enter  PF 3: Exit  PF 13: Help




* Data Field Pre-Filled By Software

AMS 039

REQUISITION FOLLOWUP

Document Number: ____-____

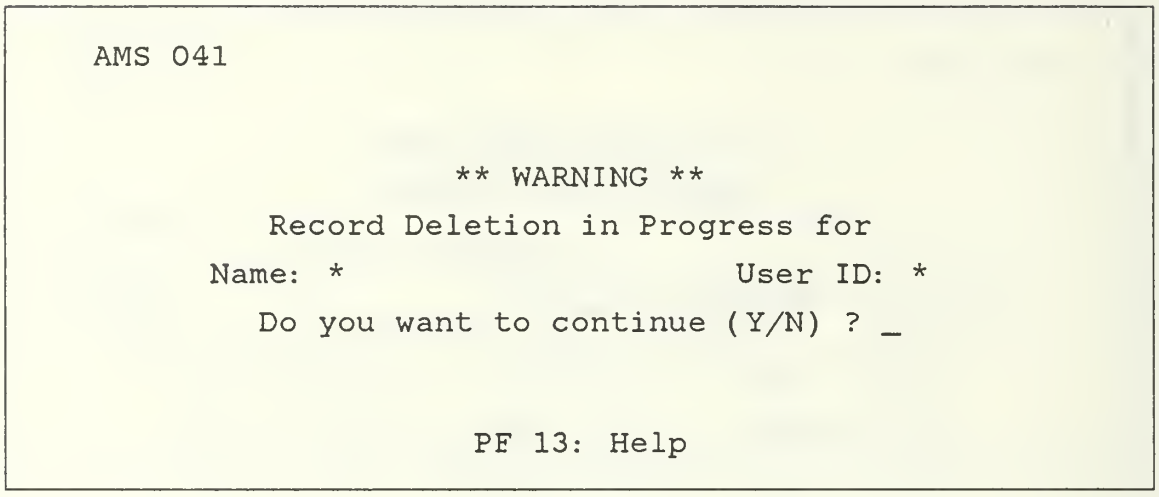
PF 1: Enter  PF 2: Quit  PF 13: Help

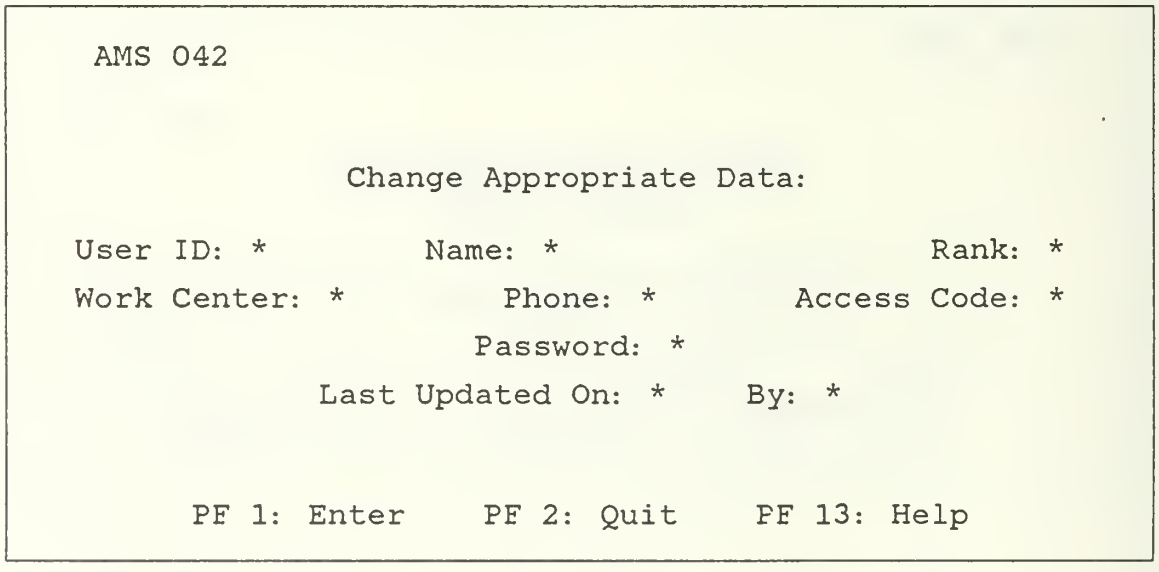
AMS 040

REQUISITION CANCELLATION

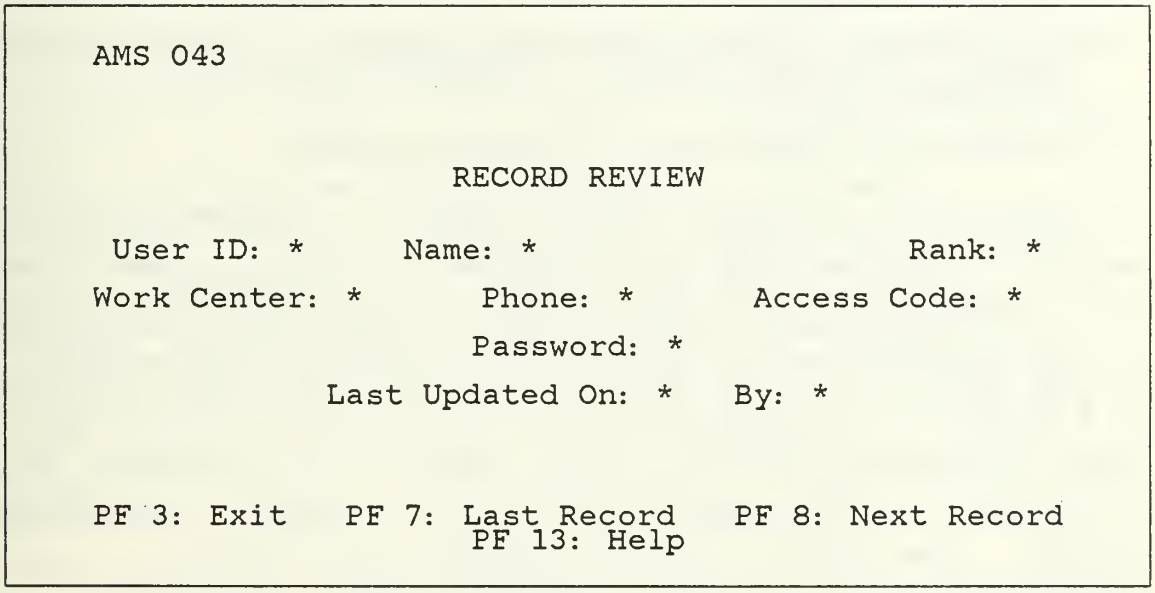
Document Number: ____-____

PF 1: Enter  PF 2: Quit  PF 13: Help




* Data Field Pre-Filled By Software


AMS 043
RECORD REVIEW
User ID: *
Name: *
Rank: *
Work Center: *
Phone: *
Access Code: *
Password: *
Last Updated On: *
By: *
PF 3: Exit
PF 7: Last Record
PF 8: Next Record
PF 13: Help

# APPENDIX F
# Detailed Functional Description

*Section 1. General.*

*1.1 Purpose of the Functional Description.* This detailed functional description for the Ammunition Management System (AMS) Shipboard Data System (SDS) is written to provide:

a. The system requirements to be satisfied which will serve as a basis for mutual understanding between the user and developer.

b. Information on performance requirements, preliminary design, and user impacts, including fixed and continuing costs.

c. A basis for development of system tests.

*1.2 Project References.* The AMS SDS is a proposed software program to automate the present manual activities associated with shipboard ammunition inventory management and reporting at the end-user level. This project serves as the subject of a graduate paper by LCDR R.B. Alderman, SC, USN of the Naval Postgraduate School. The intent of this research is to conduct the preliminary design of a shipboard application program. The design approach incorporates software engineering principles and demonstrates the methodology involved.

*1.3 Terms and Abbreviations.*

Ammunition Management System (AMS)

Ammunition Transaction Report (ATR)

Conventional Ammunition Integrated Management System (CAIMS)

Department of Defense (DOD)

Military Standard Requisitioning and Issue Priority System (MILSTRIP)

Shipboard Data System (SDS)

Shipboard Nontactical ADP Program (SNAP)

Sonobouy Transaction Report (STR)

*Section 2. System Summary.*

*2.1 Background.* Ammunition management aboard ship presents a critical challenge both in terms of the manual effort required and the resulting impact on operational readiness. Such tasks as requisitioning, status tracking, expenditure reporting and inventory management, as mandated by numerous shore activities and Fleet Commanders, represent a significant amount of administrative burden to the afloat sailor. The potential to reduce this burden through automation exists both in the realm of standardized shipboard nontactical ADP programs as well as through the use of relatively inexpensive microcomputers as a stand-alone application. The requirement to automate ammunition management is valid. However, due to present CDA resource levels and priorities established by program and functional software sponsors, such a capability is not presently available for SNAP.

The AMS SDS is one means proposed to fill this void. Another effort in this area includes the Fleet Optical Scanning Ammunition Marking System (FOSAMS).

*2.2 Objectives.* This research is limited to defining the software requirements of the end-user; that is, the software necessary to automate and support ammunition inventory management and reporting at the shipboard level. Accordingly, the unique requirements of ammunition load list management, as in the case of ammunition stores ships, is

not addressed. In addition, this effort attempts to develop an automated capability within the existing reporting structure (ie.: communications procedures and formats) and not attempt to design an independent system for this purpose.

The functional requirements produced by this study will provide the necessary guidance to CDA systems analyst personnel involved in design development of an ammunition management system. In addition, it is anticipated that general release of this report will further understanding and support for this application.

*2.3 Existing Methods and Procedures.* Ammunition inventory management and reporting procedures are established in [Ref. 2] with specific guidance promulgated by Fleet and Type Commanders. These procedures outline the local recordkeeping requirements necessary to support inventory management at the afloat end-user level. In addition, MILSTRIP requisitioning and transaction reporting procedures are established to provide the necessary external interface capability.

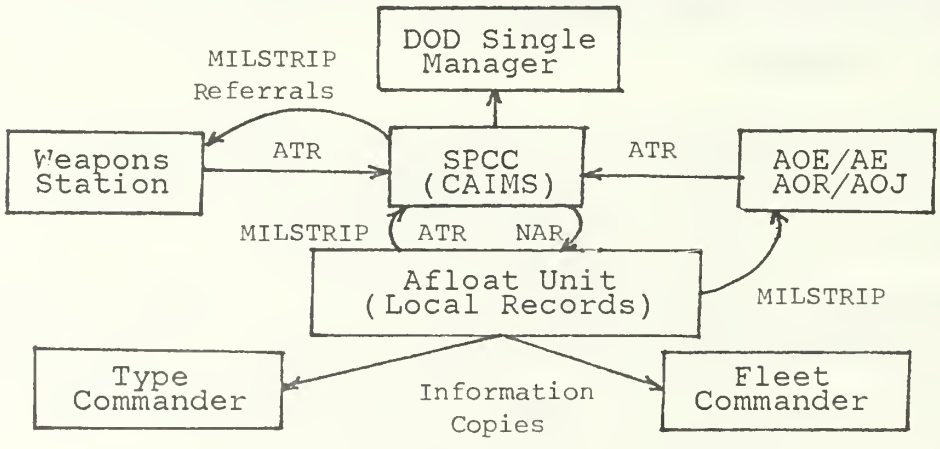



Figure F.1
External Data Flow

As ammunition management is centrally located at the DOD single manager, field input to this system is vital to provide accurate and timely system status information. For the Navy, this interface occurs at the wholesale level between the CAIMS and the DOD single manager for conventional ammunition, the Army. Retail level inventory status is reported to the CAIMS by naval activities as transactions occur. Supply documents, such as MILSTRIP requisitions, followups and cancellations, complete the necessary external interface. This data flow is depicted in Figure F.1 for external transactions. Figure F.2 depicts the manual data flow internal to the ship.

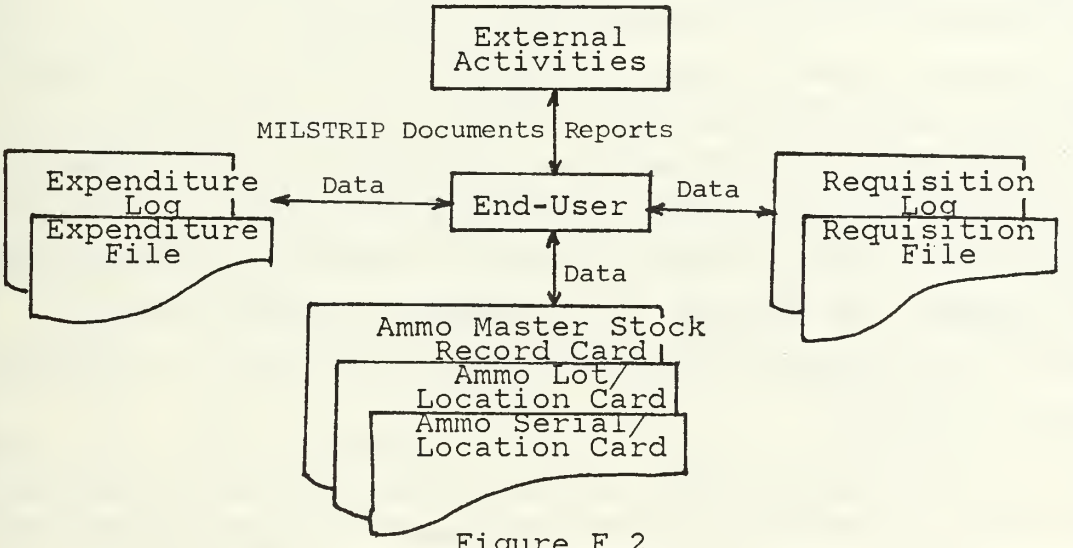



Figure F.2
Internal Manual
Data Flow

*2.4 Proposed Methods and Procedures.* The AMS SDS is designed to replace the current manual system with an interactive, menu-driven system. In general, this system will:

1. Automate the present manual file maintenance and recordkeeping effort at the end-user level.

2. Generate automated MILSTRIP documents and transaction reports.
3. Provide other automated products for use by management level personnel such as stock status listings.
4. Enable direct, online query of data for nonstandard requests by management level personnel.

The Initial Operating Capability (IOC) is designed to replace current procedures by automation, not enhance or replace an existing automated system. The automated internal data flow is depicted in Appendix A. External data flow remains unchanged as depicted in Figure F.1.

*2.4.1 Summary of Improvements.* A principal savings of the proposed system will be the reduction of the administrative burden associated with ammunition management. It is anticipated that this benefit will be realized by eliminating certain manual records, which will now be kept in online files; and the elimination of the requirement to prepare certain manual reports. In addition, the attendant increase in data accuracy and timeliness of data availability, provided by validation features of the system, will enhance the quality of the end-user/CAIMS interface. Specific improvements are presented in Section 3, Detailed Characteristics.

*2.4.2 Summary of Impacts.* Two primary areas are identified which will be affected by the implementation of the AMS SDS. These are the impact on the ship's internal organization and operation. The affects of these impacts are detailed in the following paragraphs.

*2.4.2.1 Equipment Impacts.* As the AMS SDS is a new system, no equipment upgrades or change-outs are required. The flexible nature of its design will permit incorporation in standard ADP hardware configurations such as SNAP, as well as operating as a stand-alone microcomputer application. Equipment requirements and demands on the shipboard

environmental quality are satisfied by readily available hardware and conditions. For these reasons, shipboard equipment impacts are considered minimal.

Shore establishment equipment impacts are considered negligible. Since the automated products produced by the AMS SDS duplicate their manual counterparts, no compatibility problems are expected.

*2.4.2.2 Software Impacts.* Software impacts are determined to be minimal. The structured architecture and menu scheme of the AMS SDS will permit its integration in the SNAP as a functional module. Since this design approach is in keeping with current project strategy, other benefits to accrue include integration of crew training (user and maintainer), maintenance and logistic support.

*2.4.2.3 Organizational Impacts.* The organizational impacts of the AMS SDS will be minimal. The user community for this application is narrowly defined to those divisions having cognizance over onboard ammunition stock. User access may be further limited to actual records keeping and supervisory personnel. Finally, training requirements are minimized by the direct replacement of manual forms with automated products. This precludes the necessity of having to retrain personnel on new procedures, and allows concentration on operator training.

*2.4.2.3.1 Organizational Impact in the Shipboard Environment.* The major organizational impact of the proposed system will be the restructuring of work from manual to automated means. An additional impact will be the requirement to train ship's force in the operation of the AMS SDS. However, since this requirement can be readily incorporated in onboard training programs and will include a narrowly defined user population, the impact will be minimal.

*2.4.2.3.2 Organizational Impact in the Interfaces with the Shore Establishment.* Organizational impact in this area is negligible.

*2.4.2.4 Operational Impacts.* The proposed system will not change the operational environment of the afloat user. Established ammunition management procedures will remain in effect. The only changes to be realized by the user is the mode of report and data generation with a commensurate change in the mode of transmission.

*2.4.2.5 Development Impacts.* A major system development impact is data base initiation. Data base initiation will be supported from two sources. First, a data drawdown of the CAIMS data base will provide the necessary skeleton data structure. This data can then be augmented from local records maintained by the ship. This process is depicted in Figure 2.4.

*2.5 Assumptions and Constraints.* The AMS SDS is designed to be an integrated system of data, hardware and software. It is this approach which will allow implementation as a stand-alone system or as a subsystem in SNAP. Following initiation by external activities (to include hardware installation, checkout, data base build and validation, software load and checkout, and crew training) the system will transition to organic support. Although hardware and software configuration management will remain with the Central Design Activity (CDA), crew training, equipment maintenance (to component level) and data maintenance responsibility transfers to the individual ship.

The system will be designed and configured to run unattended and/or in an unmanned space. System operation will be conducted by ship's force personnel without augmentation. Furthermore, no data processing expertise will be required

of system operators in keeping with the menu-driven philosophy. User training will be conducted as On the Job Training (OJT) and, in accordance with the established goals for SNAP, the terminal training time will not exceed the manual training time for a given function.

System management will be conducted by a system manager who will oversee the system operation, security, data base hygiene, user access and training and further serve as the ship's point of contact with external support activities. Additional responsibilities will include diskette/tape library maintenance, formulating and implementing manual fallback procedures, and conducting system backup and recovery.

*Section 3. Detailed Characteristics.*

*3.1 Specific Performance Requirements.*

*3.1.1 Accuracy and Validity.* Input to the system will be primarily interactive, online input from the users. Provisions for interfacing with the Fleet Optical Scanning Ammunition Marking System (FOSAMS) through bar code reading capability are addressed as a secondary input source. These inputs will be validated through software checks and data field range values. Other offship sources will provide appropriate controls over their input products thereby relieving the burden on the ship to perform validation checks on this data.

The following accuracy standards will apply:

a. Mathematical calculations shall be carried to sufficient decimal accuracy to ensure proper rounding.

b. Field data accuracy will be maintained in accordance with established standards.

c. Transmitted data will be maintained under current communication standards.

*3.1.2 Timing.* A system response time design goal is three seconds or less defined as the time the "ENTER" key is depressed to when the first character of the response appears on the screen. Actual system response time is a function of:

1. The number of users on the system at a given time.
2. The specific applications (MDD listing preparation, etc.) being accessed.

Response time test criteria is dependent on the above and the particular system configuration. As per lessons learned from SNAP II operational testing, an integrated, multiuser system will exhibit response times anywhere from three to thirty seconds. Accordingly, to accurately reflect the impact of system demand and configuration constraints, a response time test matrix should be constructed by application and number of users. This matrix may also be tailored to the target SNAP configuration by including other subsystem applications (word processing, organizational maintenance, etc.). Queries and/or actions requiring multiple file accesses or temporary file builds which would legitimately require in excess of five seconds will display a screen indicating that action is in progress.

*3.2 System Functions.* The following paragraphs expand and further define the individual functions presented earlier in the summary of improvements paragraph. These functions are designated for incorporation in the IOC unless otherwise indicated.

*3.2.1 AMS SDS Environment.* The following functions are considered essential for successful implementation of the AMS SDS. They describe the operating environment of the proposed system and the system features necessary to ensure secure and reliable operation.

*3.2.1.1 System Manager Functions.* The AMS SDS will provide the features necessary for the management of proper system operation. This includes user access control, data access control as well as provisions for backup and recovery.

*3.2.1.2 Printed Reports.* The following reports will be provided to the system manager:

a. User access report.
b. Summary report of system use by work center (Future Release).
c. Summary report of system use by department (Future Release).

*3.2.1.3 On-line Displays.* The following on-line displays will be provided to the system manager:

a. User access privileges.
b. Static data.
c. Message headers
d. System security breach warning (Future Release).

*3.2.1.4 Data Security.* The system will provide for secure storage and access of data files. Individual users will be assigned an access code granting specific privileges to the level of data access and manipulation capability. In addition, the range of user access to data records will be limited to the user's designated work center. Multiuser environments will incorporate audit trails for data record transactions by appending the user's identification code and date of modification to the affected record.

*3.2.1.5 Data Integrity.* The system will provide for the screening of data upon input to ensure its correctness and completeness. Data fields will be coded as either numeric, character or alpha-numeric. Records capable of access by multiple users will be provided with an edit lock mechanism to prevent concurrent updating. A cross reference file will

be provided to ensure that input data coincides with file data (Future Release). As an example, the NALC inputted will match the allowance list number and nomenclature for that item.

*3.2.1.6 Access Security.* The system will support use access control through the use of user identification codes and passwords. The system will support a monitor which will be the entry point for all users and provide basic access security. The basic security philosophy to be employed is that a user is given authority to perform a given set of functions and that only those functions are made available to him. The menu-driven system will be tailored to the specific user by excluding those functions from the menu that are not allowed.

*3.2.1.7 On-line Aid Functions.* The system will provide an on-line user's manual which will allow the user to request aid by positioning the cursor at a data field and, by pressing the help key, view the applicable user's manual page.

*3.2.1.8 Communications Interface.* The system will be capable of generating a standard 5-level paper tape for DAAS MILSTRIP messages and MILSTRIP exception messages which is capable of being read by a radio teletypewriter. The software will preassign message date-time group and other header and trailer information in addition to the text data. The system will also be capable of generating optical character recognition (OCR) message products for compatibility with over the counter service at shore communication facilities (Future Release).

*3.2.2. Inventory Management.* The AMS SDS will provide a full range of functions in order to manage ammunition and sonobouy material inventories. These functions are outlined in the following:

*3.2.2.1 Stock Record Maintenance.* The system will allow inventory data records to be maintained by NALC and further subdivided by lot or serial number. As a minimum, these records will include:

a. Complete DOD Ammunition Code (DODAC).
b. Associated allowance list number.
c. Responsible work center code.
d. National Item Identification Number (NIIN).
e. Item Nomenclature.
f. Item cognizance symbol.
g. Unit of issue.
h. Fund code.
i. Allowance quantity, total DODAC quantity on-hand, reorder quantity, and due quantity.
j. Unit price.
k. Maintenance Due Date (MDD).
l. Shelf life information.

The system will allow transaction posting as receipts and expenditures are made. A transaction history file will be included as a log for these transactions.

*3.2.2.2 Stripping to History (Future Release).* The system will allow for the downloading of the transaction history file for possible upline submission or archiving.

*3.2.2.3 Printed Reports.* The following printed reports will be provided to assist the inventory management effort:

a. Ammunition Master Stock Record Card.
b. Ammunition Lot/Location Card.
c. Ammunition Serial/Location Card.
d. Sonobouy Location Card.
e. Maintenance Due Date (MDD) Listing.
f. Expired Shelf Life Listing (Future Release).

*3.2.2.4 On-line Displays.* The following on-line displays will be provided:

a. Ammunition Item Record.
b. Sonobouy Item Record.

*3.2.2.5 Inventory Aids.* The AMS SDS will provide appropriate aid in selection of spot inventory items to be used as a management tool in judging inventory accuracy; generating inventory aids for periodic inventory of shelf life items, high value/critical items, specific cognizance symbols or by other attribute key.

*3.2.2.6 Receipt Processing.* The system will provide for the recording of receipt of material including the following special cases: partial receipt (balance outstanding), partial receipt (balance cancelled), and gains by inventory.

*3.2.2.7 Expenditure Processing.* The system will enable the expending of material due to consumption, transfer and loss by inventory.

*3.2.2.8 Requirements Determination.* The system will provide automatic screening of allowed items. This will involve the comparison of on-hand quantities against allowance quantities with a reorder listing being produced. Automatic reorder is facilitated with the option of use intervention on a line item basis. The system will also be able to identify unserviceable items based on expired MDD or shelf life code.

*3.2.2.9 Off-ship Reports and Products.* The system will be capable of generating the following off-ship reports and products:

a. Ammunition Transaction Report (ATR).
b. Sonobouy Transaction Report (STR).
c. Maintenance Due Date/Missile Firing Extension Date (MDD/MFED) Request (Future Release).

d. DD Form 1348-1, (Expenditure Document).

e. OCR compatible shipping labels based on Logistics Applications of Automated Marking and Reading Symbols (LOGMARS) 3-of-9 Universal Product Code (UPC) per MIL-STD-1189 (Future Release).

*3.2.3 Procurement of Material.* The AMS SDS will support the procurement of material with specific functions as follows:

*3.2.3.1 Requisition File Maintenance.* The system will allow requisition data records to be maintained by document number consisting of Julian date and serial number. The date and serial number will be preassigned by the system. The capability for block management of serial numbers (ie.: 8000-8999 for requisitions, 9000-9999 for expenditures, etc.) as mandated by Fleet Commanders will also be provided. As a minimum, these records will include:

a. Document number.
b. The complete DODAC.
c. Document identifier.
d. Activity routing identifier.
e. Media and status code.
f. Order quantity.
g. Demand code.
h. Signal code.
i. Distribution code.
j. Project code.
k. Priority code.
l. Required delivery date.
m. Advice code.
n. Exception information (document identifier "AOE" only).
o. Latest status information.
p. Partial order information.

The system will allow for standard requisition file maintenance including MILSTRIP followup, modifier and

cancellation generation, and status and partial receipt posting.

*3.2.3.2 MILSTRIP Products.* The system will be capable of generating the following MILSTRIP products:

a. DAAS MILSTRIP message (requisition, followup, modifier and cancellation).
b. MILSTRIP exception message (requisition).
c. DD Form 1348, 4-part (requisition).
d. DD Form 1348, 2-part (followup, modifier and cancellation).

*3.2.3.3 Reports on Magnetic Media or in Machine Readable Form.* (See paragraph 3.2.1.8).

*3.2.3.4 Compatibility With Magnetic Media or Machine Readable Input Products.* The system will be capable of accepting automated products as input from the following sources:

a. Fleet Optical Scanning Ammunition Marking System (FOSAMS). The system will be capable of reading standard 3-of-9 bar code labeling per MIL-STD-1189 by light pen or laser device (Future Release).
b. DD Form 1348-m (mechanized). The system will be capable of reading status cards provided by supply activities by card reader device (Future Release).

*3.2.3.5 Printed Reports.* The system will be capable of generating the following printed reports relating to material procurement:

a. Outstanding Requisition Listing.
b. Pending Requisition Listing.
c. Pending Expenditure Listing.
d. Aged Requisition Listing (Future Release).
e. Expired Status Date/Requisition Listing (Future Release).

*3.2.3.6 On-line Displays.* The system will be capable of generating the following on-line displays relating to material procurement:

a. Item Ammunition Requisition Input Screen.
b. Item Sonobouy Requisition Input Screen.

*3.2.4 Utilities.* The AMS SDS will provide the following utility programs in addition to the beforementioned applications:

*3.2.4.1 Ad Hoc Query.* The system will allow the direct access to data files for the preparation of non-standard reports and requests. This will be facilitated by a data base management system (DBMS) if so equipped, or a file server.

*3.2.4.2 Backup.* The system will provide for the downloading of all files and programs to magnetic media for storage external to the computer system.

*3.2.4.3 Recovery.* The system will be capable of reloading all files and programs form magnetic media to the computer system.

3.2.4.4. Electronic Mail (Future Release). In a multiuser environment, the system will provide for the transmission and receipt of electronic text from individual users to other users, or to work center, division or departments.

*3.3 Inputs and Outputs.* The system design is based on the use of currently available manual forms for data entry. The IOC permits all data entry to be conducted at a terminal with future releases expanding this capability to include DD Form 1348-m card and bar code reading. All input data will be screened on entry for completeness and correctness. Data not passing this screen will not be accepted and a display will be provided to the user notifying him of the error.

The IOC output products will resemble their manual counterparts thereby ensuring a higher probability of user acceptance and compatibility with external activities.

*3.4 Data Characteristics.* Data will be maintained in a file structure based on the processing scheme selected (ie.: database or file processing). In addition, since the user population will not be experts in computer operation, the software must provide facilities for accessing the data, naming files, etc. This data must be available for immediate use in order to support the interactive environment.

*3.5 Failure Contingencies.* The system will not allow further processing beyond a point at which data base integrity might be lost. This integrity will be enforced by check-point and backup production. When system integrity is threatened, all further processing will be locked out until suitable check-points and backups are made. These check-points and backups may be used selectively or in total by the system to restore the data base after failure.

*Section 4. Environment.*

*4.1 Equipment Environment.* The AMS SDS will incorporate off-the-shelf hardware and component devices, "ruggedized" where possible for compatibility with the shipboard environment. A stand-alone workstation should permit the processing of all system functions at that location and permit secure processing as defined in [Ref. 31]. A proposed stand-alone configuration with estimated costs follows:

| | |
|---|---|
| Zenith model 150 microcomputer | $3,800.00 |
| 120 Character per second printer | 1,200.00 |
| Facit paper tape reader/punch | 2,700.00 |
| Total: | $7,700.00 |

In addition to the above, other hardware upgrades such as a memory expansion board and uninterruptable power supply may

also be required by the program/data size or environment. These should be addressed following selection of an implementation strategy.

*4.2 Support Software Environment.* The system will require a software support environment consisting of the following:

a. An operating system capable of supporting full system resource access by the application software and system utilities, file handling, screen handling and DBMS processing (if applicable).

b. Support software such as a file server (see paragraph 3.4.2.1). Equipment mismatches may also require a character code translation program as in the case of using a Zenith 120 microcomputer (ASCII-based) with a Facit paper tape reader/punch (Baudo-based).

c. In those installations where "run time" program code is not provided, an appropriate language compiler will be included.

d. A data base management system (if applicable).

*4.3 Interfaces.* The system will provide for two distinct interfaces as follows:

*4.3.1 Interfaces among subsystems.* The primary interface between subsystems is data sharing. This permits reduction in data redundancy and also eliminates most modification anomalies. In addition, "stamp coupling" is used whereby an access vector is passed between subsystems following initial system entry by the user. This enables the identification of the user (and the corresponding access privileges) without requiring a second log in for each subsystem entry.

*4.3.2 Interfaces With Shore Commands.* The automated products produced by the system will be of such quality as to permit their upline submission in place of manual forms.

*4.4 Security and Privacy.* The AMS SDS operation and data handling will be governed by [Ref. 31]. The system will achieve initial security certification prior to implementation and will be recertified on a regular basis thereafter. In addition, privacy restrictions will be placed on the handling of data associated with users of the system.

*Section 5. Cost Factors.*

An estimate of hardware costs for a stand-alone configuration was provided in paragraph 4.1. Hardware and software development costs for implementing the AMS as a subsystem in SNAP is dependent on prorated costs and the availability of existing compatible software and hardware in the SNAP configuration. As such, this will not be addressed here.

An estimate of software development effort can be made for a stand-alone AMS application, however, which may then be used in determining development costs. One such approach is the COnstructive COst MOdel (COCOMO) proposed by Boehm [Ref. 16]. COCOMO provides an estimate of development effort in terms of man-months instead of dollar costs. The rationale for this is as follows:

> COCOMO avoids estimating labor costs in dollars because of the large variations between organizations in what is included in labor costs . . . and because man-months are a more stable quantity than dollars, given current inflation rates . . . .
>
> In order to convert COCOMO man-month estimates into dollar estimates, the best compromise between simplicity and accuracy is to apply different average dollar per man-month figure for each major phase, to account for inflation and the differences in salary level of the people required for each phase. [Ref. 16:p. 61]

The basic effort equation for an embedded-mode software project is:

$$MM = 3.6(KDSI)^{1.20}$$

where man-months (MM) is a function of program length expressed in terms of thousands of delivered source instructions (KDSI). The "embedded-mode" classification is based on the following factors:

a. The software product must operate within tight constraints (data accuracy and security).

b. The product must operate in (is embedded in) a strongly coupled complex of hardware, software, regulations and operational procedures.

As an example, a program of 128 KDSI would require 1,216 man-months of effort calculated as follows:

$$MM = 3.6(128)^{1.20} = 1,216$$

Similar COCOMO-based models exist for estimating productivity (DSI/MM), schedule (in months) and staffing requirements. The reader is referred to [Ref. 16:pp. 74-96] for an intensive treatment of this subject.

*Section 6. System Development Plan.*

The system development plan is highly dependent on the implementation strategy selected. For this reason, it is deferred to the development phase of the software life cycle.

# LIST OF REFERENCES


1. Office of the Chief of Naval Operations, OPNAVINST 8000.13, *Conventional Ammunition Integrated Management System (CAIMS); Objectives and Policies for*, February 1, 1971.

2. Navy Ship's Parts Control Center, SPCCINST 8010.12D, *Supply Management of Ammunition*, November 2, 1984.

3. Swanson, John L., "Ammunition Management at SPCC," *Navy Supply Corps Newsletter*, v. 40, n. 7, pp. 22-25, July 1977.

4. U.S. General Accounting Office, PLRD-81-54, *The Navy Must Improve Its Accountability for Conventional Ammunition*, July 29, 1981.

5. U.S. General Accounting Office, PLRD-82-27, *DOD Has Serious Problems with Care and Maintenance of Conventional Ammunition*, February 9, 1982.

6. Naval Audit Service, Audit Report Number T20040, *Multi-Location Audit of the Management of Small Arms and Ammunition Programs in the Navy*, April 1981.

7. Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, Inc., 1982.

8. Naval Weapons Support Center, Report A064, *Functional Description (FD) for the Fleet Optical Scanning Ammunition Marking System (FOSAMS I)*, prepared by Program Analysis and Management Systems Department, CACI, Inc. Federal, Contract number N00164-84-C-0076, revised November 1984.

9. Office of the Joint Chiefs of Staff, JCS Publication 6, *Joint Reporting Structure*, v. 2: *Joint Reports*, October 1984.

10. American National Standards Institute/Institute of Electrical and Electronics Engineers, Standard 830-1984, *IEEE Guide to Software Requirements Specifications*, July 20, 1984.

11. Mills, H.D., "Software Development," *IEEE Transactions on Software Engineering*, v. SE-2, December 1976.

12. Comer, D., "Principles of Program Design Induced from Experience with Small Public Programs," *IEEE Transactions on Software Engineering*, v. SE-7, March 1981.

13. Stevens, W.P. and others, "Structured Design," *IBM Systems Journal*, v. 13, n. 2, 1974.

14. Peterson, J.L. and Silberschatz, A., *Operating System Concepts*, 2nd. Ed., Addison-Wesley Publishing Co., 1985.

15. DeMarco, T., *Controlling Software Projects*, Yourdon, Inc., 1982.

16. Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, Inc., 1981.

17. Curtis, W., "Management and Experimentation in Software Engineering," *Proceedings of the IEEE*, v. 68, September 1980.

18. McCabe, T.J., "A Complexity Measure," *IEEE Transactions on Software Engineering*, v. SE-2, December 1976.

19. Mekly, L.J. and Yau, S.S., "Software Design Representation Using Abstract Process Networks," *IEEE Transactions on Software Engineering*, v. SE-6, September 1980.

20. Halstead, M.H., *Elements of Software Science*, North Holland, 1977.

21. Reingold, E.M. and Hansen, W.J., *Data Structures*, Little, Brown and Company, 1983.

22. Sprague, R.H., Jr. and Carlson, E.D., *Building Effective Decision Support Systems*, Prentice-Hall, Inc., 1982.

23. Kroenke, D., *Database Processing*, 2nd. Ed., Science Research Associates, Inc., 1983.

24. Codd, E.F., "Is Your DBMS Really Relational?" *Computerworld*, October 14, 1985.

25. Codd, E.F., "Does Your DBMS Run by the Rules?" *Computerworld*, October 21, 1985.

26. Kent, W., "A Simple Guide to Five Normal Forms in Relational Database Theory," *Communications of the ACM*, v. 26, February 1983.

27. Naval Supply Systems Command, NAVSUP Publication 508, *Supply Management Program Standard Data Element Dictionary*, June 1, 1984.

28. Commander in Chief, U.S. Atlantic Fleet, CINCLANTFLTINST 8010.4H, *Atlantic Fleet Requisitioning and Reporting Guide*, September 25, 1985.

29. Office of the Assistant Secretary of Defense (Comptroller), Standard 7935.1-S, *Department of Defense Automated Data Systems Documentation Standards*, September 13, 1977.

30. Blanchard, B.S., *Logistics Engineering and Management*, 3rd. Ed., Prentice-Hall, Inc., 1986.

31. Office of the Chief of Naval Operations, OPNAVINST 5239.1A, *Department of the Navy Automatic Data Processing Security Program*, April 1, 1985.

32. Department of Defense, DODD 5000.3, *Test and Evaluation*, April 11, 1978.

33. Office of the Chief of Naval Operations, OPNAVINST 3960.10B, *Test and Evaluation*, August 22, 1983.

34. Chief of Naval Material, *Navy Program Manager's Guide*, 1985 Ed., December 12, 1984.

35. Commander Operational Test and Evaluation Force, COMOPTEVFOR Test Plan 657-OT-IIC, *Test Plan for CNO*

*Project 657-OT-IIC Operational Assessment of Shipboard Non-tactical Automatic Data Processing System II (SNAP II) with Applications Software*, April 17, 1984.

36. Naval Supply Systems Command, NAVSUP Publication 553, *Inventory Management*, (undated).

37. Chief of Naval Material, NAVMATINST 3000.2, *Operational Availability of Weapon Systems and Equipment; Definitions and Policy*, January 21, 1981.

38. Gourlay, J.S., "Introduction to the Formal Treatment of Testing," *Software Validation*, Elsevier Science Publishers, B.V., 1984.

39. Rand Corporation, Report R-1436-DCA, *Computer Performance Analysis: Controlled Testing*, by T.E. Bell and A.C. Shetler, April 1974.

40. U.S. General Accounting Office, GAO/AFMD-83-5, *Benchmarking: Costly and Difficult, But Often Necessary When Buying Computer Equipment or Services*, October 22, 1982.

41. Keen. P.W. and Scott Morton, M.S., *Decision Support Systems, an Organizational Perspective*, Addison-Wesley Publishing Company, Inc., 1978.

42. Nicholas, J.M., "User Involvement: What Kind, How Much, and When?" *Journal of Systems Management*, v. 36, n. 284, March 1985.

# BIBLIOGRAPHY


Doyle, J., "Bar Codes Go To Sea," *Navy Supply Corps Newsletter*, v. 49, n. 1, January/February 1986.

Gorman, H.P. and others, "Operational Availability," *Navy Supply Corps Newsletter*, v. 45, n. 1, January 1982.

Hanson, O.J., *Design of Computer Data Files*, Computer Science Press, 1982.

Meyers, G.J., *The Art of Software Testing*, John Wiley and Sons, Inc., 1979.

Naval Education and Training Command, NAVEDTRA 10186-D1, *Gunner's Mate G 1&C*, 1980.

Naval Education and Training Command, NAVEDTRA 10867-C1, *The Weapons Officer*, 1982.

Navy Maintenance and Supply Systems Office, *Integrated Functional Description for Shipboard Non-tactical ADP Program II Shipboard Data System (SNAP II SDS)*, March 30, 1981.

Office of the Secretary of the Navy, SECNAVINST 5231.1B, *Life Cycle Management (LCM) Policy and Approval Requirements for Information System (IS) Projects*, March 8, 1985.

U.S. Department of Commerce, FIPS Pub 73, *Guidelines for Security of Computer Applications*, June 30, 1980.

Yourdon, E., *Structured Walkthroughs*, 3rd. Ed., Yourdon Press, Inc., 1985.

INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22304-6145 | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93943-5002 | 2 |
| 3. | Office of the Chief of Naval Operations<br>(OP-945D4)<br>Navy Department<br>Washington, D.C. 20350-2000 | 4 |
| 4. | Computer Technology Programs<br>Code 37<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 1 |
| 5. | LCDR Barry A. Frew, SC, USN<br>Code 54FW<br>Department of Administrative Sciences<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 1 |
| 6. | Professor Thomas P. Moore<br>Code 54MR<br>Department of Administrative Sciences<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 1 |

TITLE NUMBER _____ CUSTOMER NUMBER: _____

LIBRARY: Dudley Knox Library, Naval Postgraduate School
Monterey, Ca 93943

**ROSWELL BOOKBINDING**
**LIBRARY DIVISION**
2614 NORTH 29th AVENUE
PHOENIX, ARIZONA 85009
PHONE (602) 272-9338

Binding in Everything

| | F | B | NP |
|---|---|---|---|
| CONTENTS | | | |
| INDEX | | | |
| Bind without Index | | | |

ISSUE CONTENTS

| | IN | OUT |
|---|---|---|
| Discard | | |
| Bind in Place | | |
| Gather at Front | | |
| Advertisements | | |
| Front Covers | | |
| Back Covers | | |
| 1st only | | |
| Accents | | |
| Imprints | | |

Special Instructions

ROBERT BRUCE ALDERMAN

A353

Thesis A353

| | |
|---|---|
| Buck Color | 595 |
| Print Color | C-14 |
| Trim Height | |
| Ht Inches | |
| Over Thick | |
| For Title | |
| Extra Lines | |
| Extra Coll | |
| Hand Sew | |
| Slit | |
| Rules | |
| 1st Slot No | |
| Vol Slot No | |
| Year Slot No. | |
| Call # Slot | |
| Imp Slot No | |
| Type Face | |
| Price | |
| Mending | |
| Map Pockets | |
| 2 Vols in 1 | |

| | |
|---|---|
| ACTUAL TRIM | DATE |
| SPINE | JOB |
| BOARD DIM | LOT |
| CLOTH DIM | ROUTE |
| CLOTH BIN | SEQ NO |

35

219502

Thesis
A353
c.1

Alderman
Software requirements specifications for an ammunition management system.